Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
fetch.js	1 à 30	getJson()	Envoyer une requête en **GET** et Récupérer une information en JSON	dans la console, window.getJson = getJson; getJson("../config.json") doit retourner une Promise dont le PromiseState == "fulfilled" si le fichier JSON "/config.json" comporte une erreur de syntaxe dans la console, getJson("../config.json") doit retourner une Promise dont le PromiseState == "rejected" dans la console, getJson("../config2.json") doit retourner une Promise dont le PromiseState == "rejected" + Error	ressource (locale ou distante) inacessible ou mal formatée (syntaxe JSON) attention au chemin relatif : vérifer que cela fonctionne y compris sur GitLab/Hub Pages l'URL en argument pourrait être incorrecte
fetch.js	32 à 70	postJson()	Envoyer une information en JSON en **POST** et Récupérer une information en JSON	passer une commande d'articles et vérifier qu'un identifiant de commande a été retourné par le serveur	l'URL en argument pourrait être incorrecte les données en argument incorrect ou serveur down (vérifier avec Talend API tester)
fetch.js	72 à 80	loadConfig()	La fonction doit récupérer la configuration sous forme d'objet	dans la console, window.loadConfig = loadConfig; loadConfig( ) doit retourner une Promise dont le PromiseState == "fulfilled"	"../config.json" manquant ou syntaxe JSON incorrecte attention au chemin relatif : vérifer que cela fonctionne y compris sur GitLab/Hub Pages

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
helpers.js	1 à 13	toLocalPrice()	Convertir un NOMBRE en STRING formaté comme un prix (au format local) exemple : 2125.50 => "2 125,50 €"	examiner le formatage des montants sur les pages (prix unitaire, montant par article ou montant par commande) window.toLocalPrice = toLocalPrice; toLocalPrice(2.0005) =>"2,00 E" toLocalPrice(2005) =>"2 005,00 E"	si on donne en argument : string, infinity, NaN… erreur de formatage, il faut : - espace séparant les milliers - virgule (et non pas un point) - uniquement 2 chiffres après la virgule - symbole euro
helpers.js	15 à 39	toLocalDate()	Convertir un objet DATE en STRING formaté à la française (heure de Paris)	window.toLocalDate = toLocalDate; toLocalDate(new Date(1623616333281)) => "<time datetime=\"2021-06-13T20:32:13.281Z\">dimanche 13 juin 2021, à 22h32</time>" vérifier sur la page des commandes archivées	si on ne donne pas, en argument, un objet Date date et heure inexacte, mal formatée erreur de fuseau horaire

helpers.js	41 à 64	changeUrlButKeepSearch()	Intégrer un chemin relatif (STRING) à un OBJET URL,
			en conservant la search query
				window.changeUrlButKeepSearch = changeUrlButKeepSearch;
				changeUrlButKeepSearch("../html/goto.html", new URL("http://127.0.0.1:5500/pages/accueil.html?cat=1&art=x"), false);
				=> "http://127.0.0.1:5500/html/goto.html?cat=1&art=x"
				
				vérifier sur la page les liens du header par exemple
				qui doivent conserver le "cat"
					obtenir un chemin "courts" au lieu d'un chemin "complet" (http://….)
					ce qui est à éviter : pourrait ne pas fonctionner si hébergement dans un sous dossier (comme sur Pages)
					alors qu'un chemin complet permet de conserver le protocole (http/https) et l'imbrication
					
					perte de la search string
					
					perte du hash string : pas de problème actuellement (à voir si de futur développement)

helpers.js	66 à 89	reverseObject()	faire une rotation d'un objet en inversant "l'ordre" des propriétés	window.reverseObject = reverseObject; reverseObject({a:"",A:"",b:"","B":""}) => {B: "", b: "", A: "", a: ""}	si les clefs ne sont pas alphanumériques, par exemple : reverseObject({"0":"",1:"","a":"","b":""}); => {0: "", 1: "", b: "", a: ""}
helpers.js	91 à 104	changeCaseFirstLetter()	Forcer la casse uniquement sur la première lettre de la chaine de caractères	window.changeCaseFirstLetter = changeCaseFirstLetter; changeCaseFirstLetter("abcd", true) => "Abcd" changeCaseFirstLetter("Abcd") => "abcd"	si on ne donne pas, en argument, une string

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
article.js	11 à 20	getFormatedPrice()	Convertir le prix donné par un entier (en centimes) en une chaine de caractères formatée en Euros	les prix transmis par le serveur ou stockés en localStorage sont en centimes et doivent s'afficher en Euros sous la forme "à la française" vérifier les prix unitaires sur les pages produit.html	erreur de formatage mauvaise conversion centimes => Euros
article.js	22 à 63	getCard()	Construire le HTML lié à une carte BootStrap (card) présentant chaque article sur la page d'accueil	fiche produit sur la page d'accueil vérifier les hyperliens sur les fiches, au moins 2 fiches (par ex. : la première et la dernière) - clic sur l'image - clic sur le CTA "J'ACHETE" un badge rouge, situé sur les images, doit signaler la quantité (si non nulle) d'article en question contenu dans le panier (localStorage)	que la fiche soit vide, incomplète, erronée ou ayant une interactivité non fonctionnelle lorsque l'on clique sur une image : URL incomplète ou erronée, devrait ressembler à ceci : /pages/produit.html?art=5be1ed3f1c9d44000030b061&cat=1 badge présent alors que nombre mis au panier = 0 ou badge absent alors que nombre mis au panier > 0

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
articleManager.js	8 à 36	sortBy()	Trier la liste des articles sur un critère et/ou mettre dans l'ordre croissant ou décroissant	page accueil, tester les options du menu "tri" et vérifier le résultat	aucun changement tri erroné ou dans le mauvais sens sur "des plus anciens aux plus récents" (en fait il n'y a pas de date) qui est en fait un simple "reverse" peut rencontrer un souci si les clefs sont "numériques"

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
archivesManager.js	1 à 10	saveArchive()	Enregistrer les données (une liste de commandes passées) dans le localStorage	window.saveArchive = saveArchive; saveArchive(99, {essai : {orderId: "essai"}}) a dû créer listArchive_99 contenant {essai:{orderId:"essai"}} cela écrase tout : les données de l'archive sont recréées totalement et les données précédentes sont donc perdues	si l'objet ne contient pas une propriété "orderId" (devrait fonctionner mais posera des soucis ultérieurement)
archivesManager.js	12 à 22	addToArchive	Rajouter une commande passée dans l'archive dédiée à une catégorie	window.addToArchive = addToArchive; addToArchive(99,{"orderId":"essai2"}) => listArchive_99 doit contenir {essai:{orderId:"essai"}, essai2:{orderId:"essai2"}}	si l'objet ne contient pas une propriété "orderId" (devrait fonctionner mais posera des soucis ultérieurement) au lieu de rajouter, si on perd les commandes archivées antérieurement
archivesManager.js	24 à 37	getItemFromArchive()	Extraire depuis l'archive les informations relatives à une commande	window.getItemFromArchive = getItemFromArchive; getItemFromArchive(99,"essai") => {orderId: "essai"} getItemFromArchive(99,"wrong") => {} getItemFromArchive(100,"essai") => {}	la commande n'a pas été archivée la commande n'appartient pas cette catégorie
archivesManager.js	39 à 48	getArchiveNumber()	Retourner **le nombre de commandes** archivées pour une catégorie	window.getArchiveNumber = getArchiveNumber; getItemFromArchive(99) => 2 getItemFromArchive(100) => 0	il n'y a pas d'archive identifiée avec cet identifiant de catégorie
archivesManager.js	50 à 62	getArchive()	Récupérer une archive stockée sur le localStorage qui regroupe les commandes passées rattachées à une catégorie	window.getArchive = getArchive; getArchive(99) => {essai:{orderId:"essai"}, essai2:{orderId:"essai2"}} getArchive(100) => {}	il n'y a pas d'archive identifiée avec cet identifiant de catégorie
archivesManager.js	64 à 71	deleteArchive()	Effacer complètement une archive du localStorage dédiée à un identifiant catégorie spécifié	window.deleteArchive = deleteArchive; deleteArchive(99); devrait retirer le Archive_99	ne change rien au localStorage ou efface une autre entrée

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
cartManager.js	1 à 9	saveCart()	Enregistrer les données (une liste d'articles mis au panier) dans le localStorage	window.saveCart = saveCart; saveCart(200, {article1: {option1: 1, option2: 2 }}) a dû créer listCart_200 {"article1":{"option1":1,"option2":2}} écrase tout : les données du panier sont recréées totalement	création/modification d'un autre panier (autre catégorie) suite à la mise au panier d'un **premier** article : le panier ne se crée pas dans le localStorage ou bien ce panier est bien enregistré mais vide ou incomplet
cartManager.js	11 à 26	updateCart()	Remplacer dans le panier, les quantités définies (pour chaque option personnalisable) concernant un article	window.updateCart = updateCart; updateCart(200, "article1", {option1: 10, option2: 20 }) a dû modifier listCart_200 {"article1":{"option1":10,"option2":20}} updateCart(200, "article2", {option1: 10, option2: 20 }) a dû créer une nouvelle entrée dans listCart_200 qui doit valoir {"article1":{"option1":10,"option2":20}}, "article2":{"option1":10,"option2":20}	création/modification d'un autre panier (autre catégorie) suite à la mise au panier d'un second article : - aucune donnée n'est rajoutée dans le panier dans le localStorage - ou ces données sont incomplètes ou erronnées - ou cette mise à jour est OK mais ce qui existait auparavant a été perdu suite à la modification des quantités d'un article déjà mis au panier : - aucune modification dans le panier dans le localStorage - ou les quantités des autres options de cet article ont été perdues - ou cette mise à jour est OK mais les données concernant les autres articles ont été perdus
cartManager.js	28 à 42	getItemFromCart()	Extraire depuis le panier les informations relatives à un article	window.getItemFromCart = getItemFromCart; getItemFromCart(200, "article1") => {option1: 10, option2: 20} pour esayer un article non enregistré getItemFromCart(200, "article3") => {}	article non enregistré ou n'appartenant pas à cette catégorie ne trouverait pas l'article pourtant bien enregistré dans le panier retournerait les informations concernant un autre article
cartManager.js	44 à 59	getQuantityFromCart()	Calculer le nombre total d'articles contenus dans le panier, pour une référence (id) donnée et quelle que soit l'option de personnalisation	window.getQuantityFromCart = getQuantityFromCart; getQuantityFromCart(200, "article1"); => 30 pour esayer un article non enregistré getQuantityFromCart(200, "article3"); => 0	article non enregistré ou n'appartenant pas à cette catégorie retournerait une quantité fausse (0 ou celle d'un autre article, ou ne ferait pas l'addition des différents quantités par option…)
cartManager.js	61 à 75	getGlobalQuantityFromCart()	Calculer le nombre total d'articles contenus dans le panier, Toutes références (id) et options de personnalisation confondues	window.getGlobalQuantityFromCart = getGlobalQuantityFromCart; getGlobalQuantityFromCart(200) => 60 pour essayer avec une catégorie pour laquelle il n'y a pas de panier getGlobalQuantityFromCart(300) => 0	une catégorie pour laquelle il n'y a pas de panier retournerait une quantité fausse (0, ou ne ferait pas l'addition des différents quantités par option ou par article…)
cartManager.js	77 à 89	deleteItemFromCart()	Supprimer dans le panier les informations relatives à un article	window.deleteItemFromCart = deleteItemFromCart; deleteItemFromCart(200,"article2") devrait effacer l'entrée "article2" du localStorage	une catégorie pour laquelle il n'y a pas de panier article non enregistré ou n'appartenant pas à cette catégorie effacerait ou viderait tout au lieu d'effacer uniquement les données d'un seul article dans le panier
cartManager.js	91 à 104	getCart()	Retourner le panier (stocké en localStorage) lié à une catégorie	window.getCart = getCart; getCart(200) => {"article1":{"option1":1,"option2":2}}	retournerait rien alors que les données existent bien sur le localStorage ou retournerait les données d'un autre panier (autre catégorie)
cartManager.js	106 à 113	deleteCart()	Effacer complètement le panier du localStorage dédié à un identifiant catégorie spécifié	window.deleteCart = deleteCart; deleteCart(200) a dû effacer Cart_200 du localStorage	non exécution : l'entrée est toujours présente en localStorage ou effacerait/viderait les données d'un autre panier (autre catégorie)

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles

controllers.js	6 à 16	getParamsFromUrl()	Extraire la valeur du paramètre recherché depuis l'URL courante	window.getParamsFromUrl = getParamsFromUrl;
				getParamsFromUrl("cat");
				=> doit retourner la valeur du paramètre "cat" présente dans la search string de l'URL
					si on donne "" en argument ou si le parmètre n'existe pas : retournera null
					(voir le cas où l'URL contiendrait plusieurs fois le même paramètre)
					
					retourne une valeur incorrecte par rapport à l'URL

controllers.js	18 à 30	updateBtnHref()	Mettre à jour l'attribut href d'un élément HTML	http://127.0.0.1:5500/pages/accueil.html?cat=1 window.updateBtnHref = updateBtnHref; updateBtnHref("#home > a", "/test.html") document.querySelector("#home > a").href => "http://127.0.0.1:5500/test.html?cat=1"	si le sélecteur n'est pas résolu, n'est pas trouvé dans la page si l'élément HTML est résolu mais n'a pas d'attribut href

controllers.js	32 à 53	updateHeader()	Mettre à jour le HEADER
			(routine pour lancer la màj des hyperliens et des badges)
				window.updateHeader = updateHeader;
				changer de catégorie et afficher les badges avec les quantités propres à chaque catégorie
				updateHeader(2,false);
				updateHeader(2,true);
				updateHeader(4)
				
				vérifier les liens sur les 3 icones : devraient conserver la search string
					les quantités ne changent pas
					
					les hyperliens sur les icones perdent la search string qui est présente dans l'URL courante

controllers.js	55 à 82	updateNotification()	Mettre à jour le badge	window.updateNotification = updateNotification; updateNotification("cartNotification",100); => doit afficher 99+ updateNotification("cartNotification",0); => doit disparaitre updateNotification("archiveNotification",30,true); => doit afficher 30 avec une animation updateNotification("archiveNotification",-30); => doit disparaitre	un badge reste affiché alors qu'il contient 0 ou un nombre négatif un badge contenant > 99 devrait afficher 99+ pas d'animation alors que sollicité (ou l'inverse) repose sur des classes BootStrap (invisible)
controllers.js	84 à 103	checkCategory()	Vérifie la validité du numéro de catégorie par rapport à la configuration	window.checkCategory = checkCategory; checkCategory(""); => NaN checkCategory(-1); => unknown checkCategory(0); => OK checkCategory(1); => OK checkCategory(2); => OK checkCategory(3); => unknown	si l'argument n'est pas un nombre ou un nombre mais négatif ou >= nombre de catégorie l'entrée "categories" est une liste qui serait mal "numéroté" : l'id de chaque catégorie doit être == à son index (en commençant par 0)
controllers.js	105 à 114	redirect()	Redirige sur la page d'accueil de la catégorie sollicitée	window.redirect = redirect; redirect(0); => recharge la page d'accueil avec les ours redirect(1); => recharge la page d'accueil avec les caméras redirect(2); => recharge la page d'accueil avec les meubles redirect(3); => page d'accueil avec un message d'erreur "arcticles introuvables"	que la page ne se recharge pas
controllers.js	116 à 125	removeElement()	Retirer (DOM) l'élément HTML défini par un HTML ID	window.removeElement = removeElement; removeElement("home"); => l'icone "home" doit disparaitre cet action est utilisée sur la page confirmation.html lorsque l'on referme les détails d'une commande archivée (en localStorage)	si le HTML id n'existe pas sur la page s'il existe mais ne disparait pas aucun changement ou bien c'est un élément parent ou un élément enfant qui disparait
controllers.js	127 à 165	categorySelector()	Créer le menu déroulant pour basculer de catégorie (situé dans le footer)	sur chaque page, le menu doit être complet : - nom des catégories - hyper lien vers la page d'accueil avec la bonne searchstring - la page actuelle doit être "disabled" (class, tabindex et aria-disabled)	menu vide ou incomplet hyperliens inopérants ou erronés page actuelle ne serait pas "disbabled"
controllers.js	167 à 185	mainDisplay()	Fonctions communes pour l'affichage des pages : mise à jour du header et du menu déroulant du footer	pages/accueil.html doit être redirigé vers pages/accueil.html?cat=0 le header et footer doivent être complets (badges + drop down menu) et hyperliens opérationnels (les 3 icones du header et drop down menu du footer)	pas de redirection vers la page d'accueil définie par défaut (cat=0) lorsque l'url n'a pas d'argument "cat" si l'argument n'est défini, n'est pas un nombre ou est un nombre négatif, devrait alors charger la page d'accueil cat=0 le header et footer non complets (badges manquants + drop down menu vide) et hyperliens non opérationnels ou erronés (icones et drop down menu)
controllers.js	187 à 195	updateTitle()	Mettre à jour le titre principal de la page <h1>	aller sur une page d'accueil window.updateTitle = updateTitle; updateTitle("Tests") => la titre devrait changer en "Nos tests"	le titre ne change pas la lettre initiale fournie en argument reste en majuscule

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
accueil.js	7 à 28	main()	La fonction doit charger la page de manière complète	http://127.0.0.1:5500/pages/accueil.html?cat=1 doit s'afficher a minima : - les 3 menus (présentation, tri et changer de catégorie) - les badges panier et commandes archivées (si non vides)	un lien erroné sur l'une des 3 icones du header un badge non affiché alors que valeur ≠ 0 un badge affiché alors que valeur = 0 menu pour changer de catégorie incomplet, vide, incorrect ou les liens y sont incorrects ou non fonctionnels

accueil.js	30 à 63	loadCategory()	La fonction doit récupérer les données sur tous les articles
			d'une seule catégorie (id : de 0 à 2)
				dans la console, window.loadCategory = loadCategory; loadCategory(1);
				doit retourner une Promise dont le PromiseState == "fulfilled"
				
				dans la console, loadCategory(4);
				doit retourner une Promise dont le PromiseState == "rejected" + Error
				
				si le serveur backend est coupé, doit afficher un message d'excuse
					arguments erronés : "string", <0 ou >3
					(devraient être tolérés "0", "1" et "2")
					
					serveur back end coupé ou données erronées
					tester GET /api/cameras/

accueil.js	65 à 111	displayCategory()	Affiche la liste des produits	http://127.0.0.1:5500/pages/accueil.html?cat=1
				afficher tous les produits en respectant l'ordre (tri) et la présentation définis
				
				puis, dans la console, window.displayCategory = displayCategory;
				displayCategory("",1);
				doit dynamiquement remplacer la liste des produits par un message d'excuse
				
				essayer plusieurs combinaisons de choix de tri et de présentation
					liste de produits ne s'affiche pas,
					
					non respect des choix de tri et de présentation
					
					serveur back end coupé ou données erronées : tester GET /api/cameras/
					
					changer 1 des 2 choix (tri/présentation) neutraliserait l'autre choix

accueil.js	113 à 123	changeShowListener()	Changer le style de présentation : mode liste ou mode images	actionner les différentes options proposées dans le menu - image : les descriptions devraient se masquer et les cards devaient être plus petites (moins larges) - liste : les descriptions devraient réapparaitre tester le responsive	aucun changement ou présentation erronée serveur back end coupé ou données erronées : tester GET /api/cameras/

accueil.js	125 à 154	changeShow()	Changer le style de présentation : mode liste ou mode images	dans la console : window.changeShow = changeShow;
				changeShow("images");
				puis changeShow("list");
					aucun changement ou présentation erronée
					
					serveur back end coupé ou données erronées : tester GET /api/cameras/

accueil.js	156 à 174	changeSortListener()	Changer le tri : critère et sens (croissant / décroissant)	actionner les différentes options proposées - par "date" (supposition) croissant/décroissant - par prix unitaire croissant/décroissant - par nom (alphabétique) croissant/décroissant	aucun changement, ou tri erroné serveur back end coupé ou URL mal formatée : tester GET /api/cameras/

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
produit.js	7 à 24	main()	La fonction doit charger la page de manière complète	http://127.0.0.1:5500/pages/produit.html?art=5be1ed3f1c9d44000030b061&cat=1 doit afficher **a minima** - le menu (changer de catégorie) dans le footer - les badges panier et commandes archivées (si non vides) si le serveur backend est coupé, doit afficher un message d'excuse	un lien erroné sur l'une des 3 icones du header un badge non affiché alors que valeur ≠ 0 un badge affiché alors que valeur = 0 menu pour changer de catégorie incomplet, vide, incorrect ou les liens y sont incorrects ou non fonctionnels pas de message d'excuse si serveur back-end coupé

produit.js	26 à 59	loadProduct()	Récupérer depuis le backend les données relatives à un article précis	dans la console, window.loadProduct = loadProduct;
				loadProduct(1,"5be1ed3f1c9d44000030b061");
				doit retourner une Promise dont le PromiseState == "fulfilled"
				
				loadProduct(4,"5be1ed3f1c9d44000030b061");
				doit retourner une Promise dont le PromiseState == "rejected" + erreurs
					article id inexistant (ou vide) ou qui n'appartient pas à la bonne catégorie
					
					serveur back end coupé ou URL mal formatée :
					tester GET /api/cameras/5be1ed3f1c9d44000030b061

produit.js	61 à 98	displayProduct()	Afficher les données relatives à un article précis	window.displayProduct = displayProduct;
				displayProduct(["",1]);
				doit provoquer le remplacement de la fiche produit par un message d'excuse
				
				puis un refresh de la page doit rétablir la fiche au complet
					fiche produit ne s'affiche pas,
					qu'elle soit vide, incomplète, erronée ou ayant une interactivité non fonctionnelle
					
					serveur back end coupé ou URL mal formatée
					tester GET /api/cameras/5be1ed3f1c9d44000030b061

produit.js	100 à 150	htmlBuidingForArticle()	Construire le code HTML pour le contenu de la fiche d'informations	affichage complet de la fiche d'informations : nom, prix unitaire, image, description toutes les options de personnalisation doivent être présentes avec l'intitulé en français (propre à la catégorie) et la marque du pluriel/singulier doit indiquer le nombre maximal d'articles commandables (par option) prix total correct (valeur et présentation) cliquer sur l'icone "retour" doit faire revenir sur la page précédente tester le CTA pour aller "VOIR MON PANIER"	fiche produit ne s'afficherait pas, qu'elle soit vide, incomplète, erronée ou ayant une interactivité non fonctionnelle intitulé de l'option personalisable ou quantité maximale commandable seraient absents (proviennent du fichier de configuration) non conformité HTML les prix (unitaire et total) ne seraient pas affichés "à la française" (3 099,00 €) liens "retour" ou VOIR MON PANIER non fonctionnels
produit.js	152 à 200	htmlBuidingForOptionsMenu()	Construire le code HTML **du drop-down menu** qui permet de choisir une option de personalisation et de l'icone shopping-cart marqué d'un + (incrémentation)	cliquer sur "choisissez..." et toutes les options disponibles doivent apparaitre Elles sont spécifiques à chaque article l'option selectionnée ne doit pas changer entre les clicks sur le shopping-cart + arrivé au maximum commandable (dépend de la catégorie) : l'option doit être grisée et ne peut plus être sélectionnée, le menu doit revenir sur "choisissez... "	options incorrectes : aucune, incomplète ou d'un autre article après une incrémentation (action sur licone de shopping cart ayant un +) la sélection aurait changée (alors que la quantité maximale n'est pas atteinte) arrivé au maximum commandable (dépend de la catégorie) : l'option ne serait pas grisée ni désactivée le menu ne serait pas revenu sur "choisissez... "
produit.js	202 à 248	addListenerToCartIncrement()	Rajoute un comportement sur l'icone "Panier" **marquée d'un +** qui permet de rajouter 1 article avec l'option sélectionnée dans le panier (localstorage)	choisir une option puis cliquer plusieurs fois sur l'icone du shopping-cart avec un + (l'option préselectionnée ne doit pas changer entre les clicks sur le shopping-cart) - le localStorage cart_1 (_2 ou _3) doit se mettre à jour ou être créé - on doit voir le badge du panier s'incrémenter (avec une animation) - les quantités doivent se mettre à jour dans le formulaire - le prix total doit se mettre à jour selon la quantité totale **arrivé au maximum,** les clicks sur le shopping-cart ne doivent plus agir (ni incrémentation, ni animation sur le badge... ni modification en localStorage) par contre le menu doit revenir sur "Choisissez..."	les quantités ne s'incrémentent pas dans le localStorage ou sur la mauvaise option les quantités ne s'incrémentent pas dans le formulaire ou sur la mauvaise option que l'on puisse dépasser le maximum autorisé le badge ne s'incrémente pas le prix total ne se mette pas à jour
produit.js	250 à 294	htmlBuidingForOptionsForm()	Construire le code HTML **du formulaire** qui permet de saisir les quantités pour chaque option et de l'icone shopping-cart marquée d'une flèche vers le bas (mise à jour de toutes les quantités pour cet article)	les quantités proposées comme valeurs par défaut doivent être celles enregistrées dans le panier (localStorage) ou 0 si article/option absent du panier et l'icone du shopping cart doit être disabled (grisée) définir des quantités, **enregistrer** (icone shopping cart avec la flèche) puis recharger la page : les quantités enregistrées doivent être proposées comme nouvelles valeurs par défaut on ne doit pas pouvoir allez en-deça de 0, ni au-delà de la quantité maximale commandable par option (du moins en agissant par les flèches ou les touches +/-)	manquerait un champ de saisie valeur par défaut absente (vide) ou différente de celle enregistrée en localStorage en agissant sur les flèches (ou les touches +/- en simulant un smartphone) que l'on puisse aller en-deça de 0 ou au-delà de la quantité maximale commandable ou que l'on puisse saisir des lettres
produit.js	296 à 340	addListenerToCartUpdate( )	Rajoute un comportement sur l'icone "Panier" **marquée d'une flèche vers le bas** et sur les champs de saisie qui permet de mettre à jour le panier en fonctions des quantités précisées dans le formulaire	lorsque l'on clique sur l'icone "mettre à jour le panier" - le localStorage cart_1 (_2 ou _3) doit se mettre à jour ou être créé - le badge sur l'icone shopping-cart du Header doit se mettre à jour (disparaitre si le panier est vide pour tous les articles) - le prix total doit être mis à jour - le drop down menu doit être mis à jour (option désactivée si on a atteint le maximum)	à la mise à jour : - localStorage ne se met pas à jour - le badge sur l'icone shopping-cart du Header ne se met pas à jour - prix total incorrect sur la valeur ou sur la forme - le drop down menu ne serait pas affecté si une des valeurs a atteint la quantité maximale

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
produit.js	342 à 379	updateTotalPrice( )	Valide et corrige, si nécessaire, les quantités, détecte et signale les modifications, active/désactive l'icone "Panier" marquée d'une flèche vers le bas et met à jour le prix total pour l'article courant	changer en saisissant au clavier + et - qui devraient être retirées immédiatement de même pour les "leading zeros" si la champ a été vidé, il devrait se rétablir à zéro de même une quantité saisie au-delà du maximum doit être ramenée à ce maximum modifier une valeur par rapport aux valeurs d'origine : le champ doit etre encadré de vert et l'icone du chopping-cart doit être enabled (en couleur au lieu de grisée) puis rétablir les quantités d'origine doit annuler tout cela le prix total doit se mettre à jour selon les quantités corrigées à chaque modification saisie	que l'on puisse saisir au clavier des signes + ou - (+3 ++3 -3 --3 3-2 3+2 3++ + - etc…) ou des "leading zero" (01, 002...) et qu'ils persistent qu'un champ puisse rester vide (sans être raméné à zéro) qu'une valeur puisse dépasser la quantité maximale (sans être raméné à ce maximum) suite à une modification de valeur : le cadre vert n'apparaisse pas ou que l'icone du shopping cart ne soit pas enabled (en couleur au lieu de grisée) si rectifier toutes les modifications de quantité n'annule pas ces effets (cadre et bouton) prix inexact ou mal formaté ("à la française")
produit.js	381 à 396	calculateTotalPrice( )	Calculer le prix total en fonction du nombre d'articles choisis dans chaque option	indiquer au moins 1 article dans une des options, puis dans la console : window.calculateTotalPrice = calculateTotalPrice; calculateTotalPrice(10000); devrait mettre à jour le prix : 100 Euros x nombre total d'article	aucune modification, ou mauvais calcul ou rien (vide) ou nombre mal formaté (pas "à la française")
produit.js	398 à 424	buildZoom( )	Afficher dynamiquement un zoom sur l'image et permettre de refermer ce zoom	cliquer sur l'image : elle devrait s'afficher en plus grande taille cliquer ensuite sur le card header pour refermer ce zoom	le zoom ne s'ouvre pas, ou en se referme pas mauvaise image

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
panier.js	8 à 47	main()	La fonction doit charger la page de manière complète	http://127.0.0.1:5500/pages/panier.html?cat=1 doit afficher **a minima** - le menu (changer de catégorie) dans le footer - les badges panier et commandes archivées (si non vides) si le serveur backend est coupé, doit afficher un message d'excuse	un lien erroné sur l'une des 3 icones du header un badge non affiché alors que valeur ≠ 0 un badge affiché alors que valeur = 0 menu pour changer de catégorie incomplet, vide, incorrect ou les liens y sont incorrects ou non fonctionnels serveur back-end coupé ou données erronées : tester GET /api/cameras/ pas de message d'excuse si le serveur est coupé

panier.js	49 à 79	loadInfoForCart()	Récupérer la configuration du site
			et les données relatives à tous les articles d'une catégorie
			qui seront nécessaires
				dans la console, window.loadInfoForCart = loadInfoForCart; loadInfoForCart(1);
				doit retourner une Promise dont le PromiseState == "fulfilled"
				
				dans la console, loadInfoForCart(4);
				doit retourner une Promise dont le PromiseState == "rejected" + Error
					arguments erronés : "string", <0 ou >3
					(devraient être tolérés "0", "1" et "2")
					
					serveur back end coupé ou données erronées
					tester GET /api/cameras/

panier.js	81 à 178	displayCart()	Gère l'affichage du Panier sous forme d'une card (BootStrap)	clic sur le "card header" : doit faire un back() si serveur injoignable : affichage d'un message de warning si le panier est vide : affichage d'un message d'information constituer un panier avec plusieurs articles différents avec des panachés d'options le panier doit alors être affiché sous forme d'un tableau avec : - chaque article : son nom, son prix, sa quantité, - les TOTAUX :quantité totale et prix total - le début de chaque ligne article doit être lié à la fiche produit de l'article - les articles à quantité zéro ne sont pas affichés - les articles n'existant plus sur le serveur : message d'information et ne feront pas partie du montant ou de la commande (on peut simuler ceci en changeant l'id d'un article sur le localStorage)	pas de message d'alerte si serveur back coupé tester GET /api/cameras/ pas de message si panier vide affichage ne correspondant pas au panier du localStorage pas de message d'info sur les articles n'existant plus et/ou prise en compte de leur quantité sur les totaux (prix total et nombre total) calculs erronés (multiplications et additions) prix incorrects au niveau valeur ou forme
panier.js	180 à 198	removeArticleFromCart()	Supprimer un article de la liste	constituer un panier avec plusieurs articles différents cliquer sur la corbeille (à droite) pour supprimer une référence : - la ligne doit disparaitre - les quantités et prix doivent se mettre à jour - le badge du panier doit se mettre à jour passer la commande pour vérifier que l'article a bien été retiré de la commande en comparant avec la vue détaillée de la commande archivée	la ligne supprimée ne disparait pas prix total et quantité totale ne se mettent pas à jour le badge (icone panier) ne se met pas à jour bien que disparu visuellement, l'article supprimé resurgisse dans la commande
panier.js	200 à 296	displayForm()	Gère l'affichage du formulaire de contact, d'acceptation des CGV et envoi de la commande	nom et prénom : entre 2 et 50 caractères lettres (y compris accents…), espace et trait d'union adresse : entre 5 et 50 caractères lettres (y compris accents…), espace, trait d'union et chiffres ville : entre 2 et 50 caractères lettres (y compris accents…), espace, trait d'union et slash CP : exactement 5 chiffres email : conformité à la norme bouton rénitialiser pour tout vider	que l'on puisse saisir des caractères non autorisés sans que le champ ne soit encadré de rouge vérifier les limites de saisie par exemple le Code Postal : limité à 5 bouton "réinitialiser" ne vide pas les champs
panier.js	298 à 314	checkCart()	Activer/Désactiver le CTA "COMMANDER" **en fonction du panier** ce panier est-il vide ? en tenant compte des articles dont la référence n'existe plus sur le serveur backend	constituer un panier et remplir le formulaire (infos de "contact") le bouton CTA "COMMANDER" doit s'activer supprimer les articles ou si les articles ne sont plus disponibles (n'existent plus sur le back end) le bouton doit se désactiver (et cliquer dessus ne doit pas avoir d'action)	le bouton COMMANDER ne se désactive pas (muted) après que les articles : - soit qu'ils aient été retirés du panier - soit qu'ils n'existent plus sur le serveur back end (ancienne référence/id)
panier.js	316 à 355	checkForm()	changement de styles **en fonction du formulaire** est-il valide ?	si on respecte les règles du formulaire ; le message de consigne doit passer au vert et le bouton CTA "COMMANDER" doit être de couleur violet (primary) passer d'une situation à l'autre plusieurs fois, avec un panier vide ou non vide	formulaire complet et conforme mais le message resterait orange et bouton resterait grisé ou bien vider un champ d'un formulaire complet et conforme ne réinitialise pas le message en warning et le bouton en grisé

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
panier.js	357 à 391	sendOrderBtn()	Comportement du bouton d'envoi de la commande	vérification de la conformités des champs du formulaire **(tooltip** explicatif si non-conformité) la checkbox (acceptation des CGV) doit être cochée toutes ces conditions doivent être respectées et de plus, la quantité globale finale réelle (en soustrayant les références "introuvables") doit être > 0	si panier non vide (avec références encore valides), cliquer sur le bouton n'afficherait pas la tooltip sur un champ non conforme (par exemple un caractère interdit) ou sur la checkbox (acceptation des CGV) non cochée si panier non vide, et formulaire complet et conforme, la bouton ne lancerait pas la commande que l'on puisse lancer une commande alors que la panier a été vidé après complétion du formulaire
panier.js	393 à 467	sendCommand()	Gère la commande, envoi et récupération des données	envoi la commande au serveur back-end si bien traitée : - le panier doit être vidé (supprimé du localStorage) - la commande doit être rajoutée sur les archives en localStorage - la badge panier doit disparaitre (puisque mis à zéro) - le badge commandes archivées doit se mettre à jour (incrémentation de +1) - affiche la référence de la commande avec un hyperlien + montant total - le titre change (remerciement)	si serveur coupé tester POST (avec Talend API Tester) si communication OK : - problème de non conformité des données envoyées - ou envoyées sur une URL incorrecte (mauvais chemin) - non suppression en localStorage du panier - non enregistrement en localStorage des commandes archivées - valeur des badges (header) incorrectes ou non mis à jour (besoin de refresh la page) - le titre ne change pas - hyper lien sur la référence de la commande incorrect, devrait avoir cette forme : http://127.0.0.1:5500/pages/confirmation.html?cat=0#e134c120-c84d-11eb-9054-bf2a8a75ce78 - le montant ne s'affiche pas, ou mal, ou est erroné (incohérent avec la montant précédemment calculé)
panier.js	469 à 480	checkRealTotalQuantity()	Relève dans la page la quantité d'articles trouvés et donc réellement commandables	window.checkRealTotalQuantity = checkRealTotalQuantity; checkRealTotalQuantity() doit donner la quantité affichée sur la ligne TOTAUX	la valeur retournée ne correspond pas à celle lue sur la page

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problèmes possibles
confirmation.js	8 à 25	main()	La fonction doit charger la page de manière complète	http://127.0.0.1:5500/pages/archivel.html?cat=1 doit afficher a minima - le menu (changer de catégorie) dans le footer - les badges panier et commandes archivées (si non vides) PAS BESOIN du serveur backend sur cette page	un lien erroné sur l'une des 3 icones du header un badge non affiché alors que valeur ≠ 0 un badge affiché alors que valeur = 0 menu pour changer de catégorie incomplet, vide, incorrect ou les liens y sont incorrects ou non fonctionnels
confirmation.js	27 à 36	displayEmptyArchiveMessage()	Affiche un message signalant qu'il n'y a pas de commande archivées sur ce localStorage	se rendre sur une catégorie n'ayant pas d'archives (changer de catégorie ou sur un autre navigateur) si le badge vert (sur l'icone archives) est absent => le message ("archives introuvable") devrait s'afficher	page vide au lieu d'avoir le message
confirmation.js	38 à 79	displayArchive()	Création de cards représentant chaque commande archivée en localStorage	récupère du localStorage et affiche : - id du bon de commande tel que renvoyé par le serveur backend - totaux (montant et quantité) - la date un bouton shopping cart permet d'aficher les détails puis de refermer les détails	nombre de cards ne correspondant pas au localStorage affcihage incorrecte, incomplet ou vide date mal affichée (format) bouton shopping cart inopérant (n'ouvre pas et/ou ne referme pas les détails)
confirmation.js	81 à 170	displayOrder()	Gère le contenu et l'affichage des détails d'une commande archivée	récupère du localStorage sous forme d'un tableau le "reçu" (receipt) qui a été renvoyé par le serveur back-end : - chaque article : son nom, son prix, sa quantité, - les TOTAUX :quantité totale et prix total - le début de chaque ligne article doit être lié à la fiche produit de l'article	affichage ne correspondant pas à l'archive du localStorage ni à la commande qui a été passée (notament les quantités et les montants) que des articles à zéro ou qui n'existaient plus sur le serveur au moment de la commande soient présents ou prise en compte de la quantité totale ou le montant total calculs erronés (multiplications et additions) prix incorrects au niveau valeur ou forme
confirmation.js	172 à 185	getTotalsFromReceipt()	Calculer les totaux (prix et quantité) en exploitant les données du reçu qui a été retourné par le serveur back-end lors de la commande	afficher les détails et comparer les totaux (quantité et montant) affichés sur le résumé et sur le détail (puisque 2 méthodes différentes)	pourrait y avoir une différence si le prix n'était pas le même selon les options