Plan de test pour le site d’e-commerce Orinoco

Introduction

Ce plan de test décrira comment tester les fonctionnalités du site Orinoco. Le but de ce projet est de créer un site Web de commerce en ligne, qui **affiche** tous les produits disponibles et qui permet aux utilisateurs d’**ajouter** les articles sélectionnés au panier, avec pour objectif de remplir un **formulaire** pour soumettre leur commande.

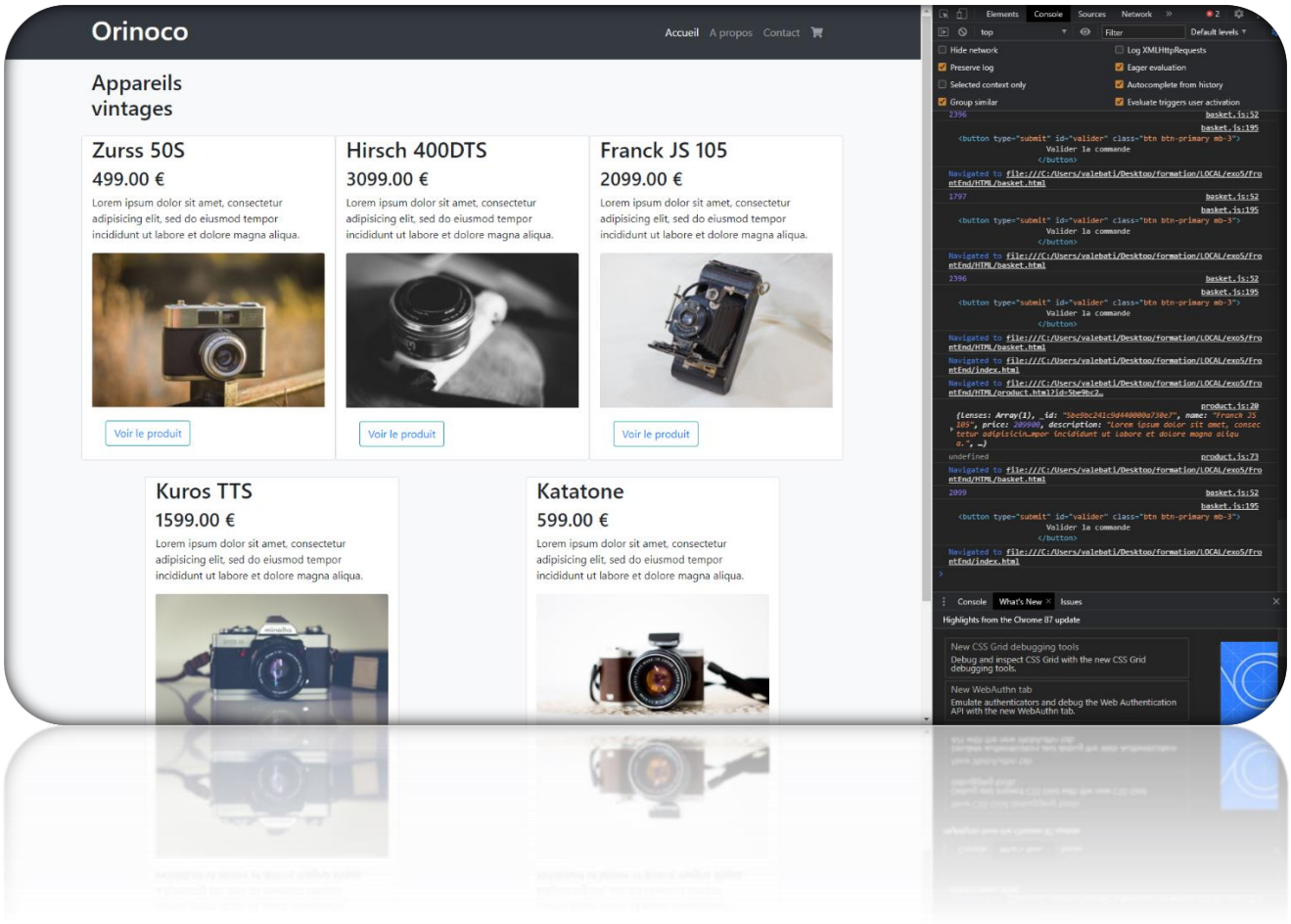
Nous rappelons que le site contient les exigences suivantes :

1. La page d’**accueil** affichera une liste de tous les produits disponibles.
2. La page d’un seul élément remplira les informations de l’élément sélectionné.
3. La page du panier affichera le ou les article(s) que l’utilisateur a sélectionné avec le total actuel et un **formulaire** pour qu’ils remplissent leurs informations ; le formulaire doit être **valide** avant de pouvoir être soumis.
4. Une page de **confirmation** qui remerciera l’utilisateur pour son achat et qui affichera un numéro de commande ainsi que le total final de son achat.

Fonctionnalités à tester

Nous pouvons tester dans un premier temps notre premier objet qui est une promesse (Promise), et qui a pour but de nous retourner le tableau de tous les éléments grâce à l’utilisation de la méthode *GET*.

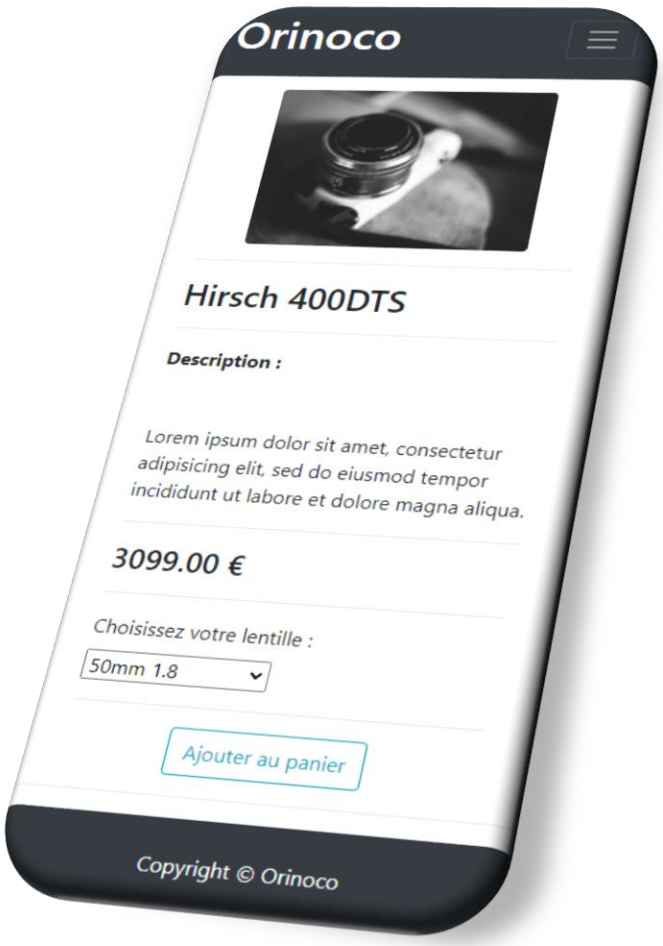
PAGE SCRIPTS.JS



- **Ligne 1 à 12** => fonction *apiRequest.onreadystatechange* doit être égale à *4* pour signaler que l'opération s'est bien terminée.
  Ensuite *apiRequest.status* doit être égale à *200* pour nous confirmer que notre requête HTTP est correcte afin de rendre l'analyse de la chaine de caractère JSON pour construire la valeur JavaScript ou l'objet décrit par cette chaine (*JSON.parse(apiRequest.response)*).
  Si la valeur de *apiRequest.status* est différent de *200*, alors nous devons recevoir un message d'erreur.

- **Ligne 20 à 59** => fonction asynchrone *makeRequest().then()* doit donc nous retourner la liste des caméras avec les infos correspondantes. La méthode *map()* nous permet de de créer un nouveau tableau avec les résultats de chaque élément *lense*.

- **Ligne 11 à 20** => fonction asynchrone *adress* doit nous retourner le résultat *json* de chaque caméra avec leurs infos que nous récupérons grâce à l'*ID* du produit. (Méthode *GET* renvoie l'élément correspondant à l'identifiant *given_id*).
  Pour vérifier le résultat, nous pouvons dans un premier temps *console.log* l'argument *URL* afin de vérifier qu'il s'agit bien du résultat attendu. Ensuite nous pouvons *console.log element* afin de vérifier qu'il s'agit bien de la caméra que nous avons sélectionné.
  Si l'argument n'est pas valide ou que la constante *elementId* contient une anomalie, l'instruction *catch* nous retournera une *alert* contenant un message d'erreur.



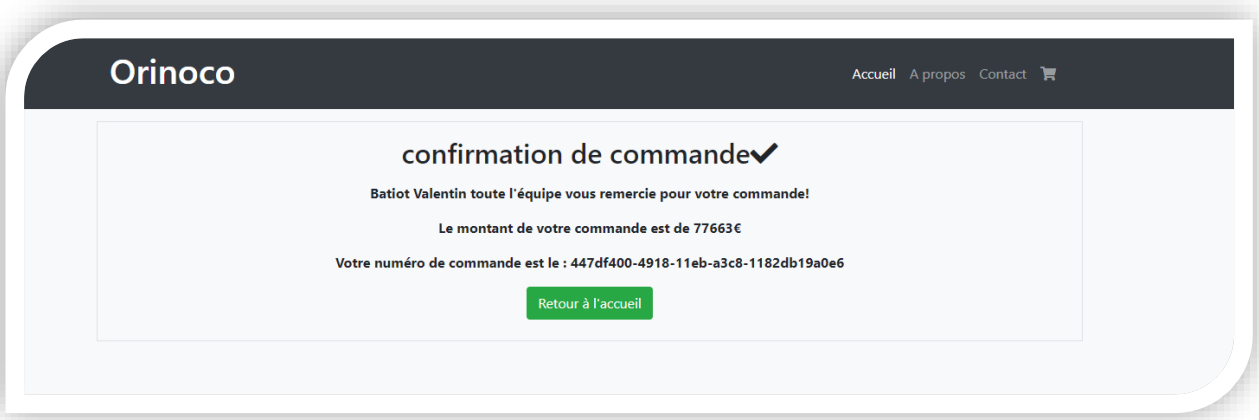
- **Ligne 65 à 73** => variable *elementDuPanier = panierAjout.find();* nous permet de récupérer dans le tableau le premier produit qui correspond à l'ID et à la lentille sélectionnée. Ces éléments sont recueillis et affichés depuis le localStorage. Nous pouvons vérifier le résultat grâce à un console.log(elementDuPanier). Il faut s'assurer que l'élément désigne bien la lentille attendue et correspondant à la caméra choisie.

- **Ligne 75 à 99** => variable *elementDuPanier* doit ajouter un nouvel item au panier s'il n'existe pas déjà, si non il doit ajouter une nouvelle quantité sur l'élément produit correspondant.
  Si le produit existe déjà mais qu'il contient une lentille différente, alors il devra créer un nouvel élément.
  Nous pouvons vérifier les différents cas possibles avec des console.log, ou en programmant directement d'autres valeurs de tests, dans le but de s'assurer de ne pas avoir qu'un seul produit pour plusieurs lentilles, ou bien plusieurs produits pour une même caméra.

PAGE BASKET.JS

- **Ligne 205 à 241 =>** fonction validData doit retourner false si les formats des différents champs ne sont pas valides, si non elle retourne true. Nous pouvons vérifier le résultat en testant la fonction avec différentes valeurs de tests. Par exemple avec moins de 3 caractères, un simple ou plusieurs espaces, des numéros pour le nom ou encore des caractères spéciaux.
  La valeur retournée pourrait être erroné (0 au lieu de 1, et 1 au lieu de 0).

- **Ligne 243 à 271 =>** asynchrone fonction donneesAttendu qui doit nous rediriger vers une autre page si tout s'est bien passé, si non elle nous retourne une erreur qui nous renvoie une alert correspondante. Nous pouvons faire un console.log de data pour vérifier que les données attendues sont correctes, si non vérifier que nous avons bien une erreur le cas échéant. Si la réponse au format JSON correspond également, alors la redirection vers la dernière page peut s'effectuer.



PAGE ORDER-CONFIRMATION.JS

- **Ligne 46 =>** fonction createTag prend 3 arguments et retourne titreConfirmation contenant la balise titre, la classe et le texte HTML du titre de la page. Nous pouvons tester le bon fonctionnement de createTag en l'appelant avec différentes valeurs de tests. La valeur retournée pourrait ne pas correspondre à du HTML.