PLAN DE TEST

Orinoco

Ce plan de test nous permettra de repérer les anomalies du site avant que les utilisateurs de notre site ne les découvrent. Il s'agit donc d'une vérification en amont des différents éléments de notre site Orinoco. Pour cela, nous utiliserons le framework « Mocha » afin de lancer des tests unitaires en syntaxe BDD sur nos fonctions Javascript.

Pour initialiser des tests côté navigateurs il va falloir créer un fichier HTML qui va être capable de faire fonctionner nos tests. Pour cela on peut utiliser la commande « mocha init <dossier> » dans notre terminal de commande.

Test N ° 1 : Vérifier la bonne exécution des produits dynamiques de la page index.html

L'utilisateur se connecte sur index.html. Les caméras s'affichent avec image, titre, prix et description.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
+ index.js (dossier Orinoco)

Lignes testées : Index.js -> ligne 10 -> function(data) ;
Nous testerons cette fonction sur le fichier test .spec.js

Résultat : Nous nous attendons à ce que notre page affiche le nombre d'articles dans le serveur. Si il n'y à pas d'articles dans le serveur, aucun article ne sera affiché dans la page index.html.

Test N ° 2 : Vérifier la bonne utilisation des url dynamiques pour la page produit.html

Quand l'utilisateur click sur une caméra, la page produit.html s'affiche avec les informations de la caméra sélectionné. Tout cela de façon dynamique bien entendu.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
produit.html + produit.js (dossier Orinoco)

Lignes testées : produit.js -> ligne 13 ->let article = function() ;

Résultat : Si nous changeons d'id produit, alors notre barre d'adresse html changera aussi en fonction de l'id. Si un produit ne possède pas d'id, alors nous seront dans la page produit.html par defaut.

Test N ° 3 : Vérifier le retour serveur

Vérifier que toutes les éventualités du retour serveur soit prisent en compte. Pour cela, nous tapons, par exemple, une autre adresse que le localhost afin de simuler un problème serveur et vérifier le message d'alerte.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
index.js (dossier Orinoco)

Lignes testées : index.js -> ligne 77 -> function (error) ;

Résultat : Un message d'Alerte apparaitra sur l'écran de l'utilisateur si le statut de onreadystateatchange est différent de 4 et que l'état du retour serveur est différent de 200. Sinon, la fonction(response) sera executée.

Test N ° 4: Contrôler les informations de l'article sur la page produit.html

Vérifier que la page produit.html affiche correctement les éléments de la caméra sélectionné sur index.html
Tout cela, de façon dynamique.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
produit.html + produit.js (dossier Orinoco)

Lignes testées : produit.js -> ligne 36 -> function(affichageProduit) ;

Résultat : La caméra s'affiche correctement avec son images, prix et nom. Tout cela, récupérer dans le serveur grâce à l'id de la caméra dans l'url dynamique.

Test N ° 5: Ajout de l'article dans le LocalStorage

Vérifier que la caméra se rajoute dans le LocalStorage lorsque nous cliquons sur le bouton « Ajouter au Panier ».

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
produit.js (dossier Orinoco)

Lignes testées : produit.js -> ligne 104 -> function(ajoutLocalStorage) ;

Résultat : Dans le LocalStorage, nous devons avoir un Objet contenant les Caméras mis au panier lorsque l'utilisateur click sur le bouton « Ajouter au Panier ». Celui-ci se mettra à jour lorsque nous rajouterons des produits dans le panier.

Test N ° 6: Message d'informations quand ajout du produit au panier

Vérifier qu'un message vient alerter l'utilisateur lorsque celui-ci ajoute un article au panier.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
produit.html + produit.js (dossier Orinoco)

Lignes testées : produit.js -> ligne 85 -> console.log("L'article "+ reflex.name + "a été ajouté au panier");

Résultat : Nous avons dans la console un message qui vient nous confirmer que l'article a bien été ajouté au panier ainsi qu'un message d'alerte dans la page produit.html

Test N ° 6: Vérification de la page Panier.html vide

Lorsque l'utilisateur n'a sélectionné aucun article et click sur la page panier, aucun article ne doit s'afficher et le montant total de la commande doit être égal à 0€. Un message « Votre panier est vide » doit aussi être affiché.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
panier.html + panier.js (dossier Orinoco)

Lignes testées : panier.js -> ligne 25 -> console.log(div.textContent);

Résultat : Nous avons dans la console le message « votre panier est vide », ainsi que dans notre page panier.html.

Test N ° 7: Vérification des articles ajouter au Panier

Lorsque l'utilisateur ajoute des articles au panier dans la page produit.html, ces articles doivent tous se retrouver dans la page panier.html.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
panier.html + panier.js (dossier Orinoco)

Lignes testées : panier.js -> ligne 31 -> Object.values(data).map((reflex) => {}

Résultat : Nous testerons ici la fonction qui nous permets, si nous avons des articles dans le localStorage, de les afficher de manière dynamique dans notre page panier.html. Si nous n'avons pas d'articles dans le localStorage, alors aucun article ne sera affiché dans notre page panier.html.

Test N ° 8: Vérification de la fonction numStr() pour le séparateur de millier

Nous devons avoir un séparateur de millier pour nos montant en €. Ainsi, le nombre 12000 s'affichera 12 000 grâce à cette fonction.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
panierDynamique.js (dossier Orinoco)

Lignes testées : panier.js -> ligne 14 -> function numStr(a, b)

Résultat : Sur l'ensemble de nos page html, nous avons bien nos nombre qui bénéficie du séparateur de millier grâce à la fonction js numStr(). Si un nombre est une centaine, une dizaine ou un simple chiffre, alors aucun séparateur de millier ne sera mis en place.

Test N ° 9: Vérification du montant total dans le panier

Nous devons avoir, dans la page panier.html, un montant total qui sera dynamique en fonction des éléments ajoutés ou retirés du panier.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
panier.html + panier.js (dossier Orinoco)

Lignes testées : panier.js -> ligne 9 -> console.log("Montant du panier actuellement: ", numStr(total), "€");

Résultat : Nous avons dans la console le montant total du panier en dynamique. Celui-ci s'ajustera en fonction des actions de l'utilisateur.

Test N ° 10: Vérification du bouton « supprimer l'article » dans la page panier.html

Nous devons vérifier si le bouton « supprimer l'article » dans la page panier.html fonctionne correctement.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
panier.html + panier.js (dossier Orinoco)

Lignes testées : panier.js -> ligne 80 -> function deleteButtons()

Résultat : Lorsque l'on click sur ce bouton, nous voyons bien que l'article concerné quitte le panier, que le montant total est mis à jour et que le LocalStorage est à jour aussi. Ce bouton ne doit s'afficher que lorsqu'un produit est dans le panier. Si aucun article est dans le panier, ce bouton et donc cette fonction ne sera pas disponible.

Test N ° 11: Vérification de la confirmation commande

Nous verrons ici comment vérifier notre envoi de donnée vers le serveur grâce à la méthode POST .

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
panier.js (dossier Orinoco)

Lignes testées : panier.js -> ligne 130 -> function(achat) ;

Résultat : Lorsque nous validons notre commande avec les éléments du formulaire ainsi que les articles, nous envoyons le montant total du panier sont au serveur. Cette fonctionnalité ne peut être utilisé que si le panier possède au moins un article.

Test N ° 12: Vérification du contrôle des entrées des formulaires

Nous vérifierons que les saisies de l'utilisateur dans nos formulaires sont bien contrôlés avant envoi de la commande au serveur.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
panier.js (dossier Orinoco)

Lignes testées : panier.js -> ligne 130 -> function(achat) ;
ligne 191 à 209 -> utilisation des expressions régulières (Regex).

Résultat : L'utilisateur ne peut indiquer des chiffres dans le formulaire nom ou prénom. Il doit indiquer une adresse mail sous un format valide…. Sinon, la saisie ne sera donc pas conforme et il devra recommencer.

Test N ° 13: Vérification de l'initialisation du LocalStorage à la fin de la commande

Nous vérifierons qu'une fois la commande passée et que l'utilisateur click sur le bouton « Retour à l'accueil du site », le LocalStorage est initialisé et que revient à 0 donc.

Documents concernés : index.html + test.spec.js (dossier test_unitaire)
order.js (dossier Orinoco)

Lignes testées : order.js -> ligne 31 -> function(retour);
Résultat : L'utilisateur revient sur la page d'accueil du site avec un panier + LocalStorage vident.
Il ne doit rien rester de sa commande passée sur le localStorage.