Planification des tests

Après avoir lu le cours d'OpenClassrooms sur les tests, quelque chose m'a choqué. Comment est-on sensé faire des tests unitaires pour du front, quand toutes nos fonctions dépendent les unes des autres ? Dépendent du « *local storage* », ou d'autres données sur d'autres pages, ou dans l'URL, etc ?

Autant faire des tests unitaires pour une application dont on peut isoler les fonctions semble logique (une application par exemple), autant faire des tests unitaires pour toutes mes fonctions dans ce projet me semble presque impossible.

On peut, en revanche, très facilement faire des tests fonctionnels dans lesquels on va analyser les éléments un par un, tout en adoptant une vision globale. C'est ce que je vais faire ici car c'est la seule chose qui me semble logique et possible de faire.

J'ai bien entendu testé mon code fonction par fonction lors de sa conception, sinon le site ne marcherait pas.

Pour toutes les pages

La bannière Orinoco doit toujours bien ramener vers la page d'accueil, tout en préservant l'état actuel du panier.

Les deux liens du footer (qui sont surtout là en tant que place-holders) doivent fonctionner sur toutes les pages, ainsi que leurs animations.

Page d'accueil

Cette page affiche juste la liste des produits de base (en l'occurrence les oursons). Il faut juste s'assurer qu'ils apparaissent correctement, avec les données correspondantes.

Le bouton « consulter le panier » doit bien mener à la page panier, que ce dernier soit vide (et il doit le montrer) ou plein, et ce quelque soit la page à partir de laquelle il a été rempli.

Chaque vignette de produit doit correctement afficher le bon produit dans la page qui suit.

Page individuelle des produits

Le bon produit doit s'afficher, mais il doit toujours s'agir du même fichier HTML qui est chargé (il ne doit pas y avoir une page par produit qui se charge indépendamment).

Le menu pour sélectionner la couleur de l'ourson doit s'afficher correctement, et avoir une ligne « choisir une couleur ». Cette ligne ne doit pas pouvoir être validée, le bouton permettant d'ajouter

l'ourson au panier doit demander qu'une couleur soit spécifiquement sélectionnée (on ne veut pas avoir dans son panier un ourson dont la couleur est « Choisissez une couleur » !).

Page panier

Le panier doit correctement afficher tous les oursons ajoutés depuis la page précédente, et ne doit donc pas se vider si l'on revient dans les pages précédentes par exemples.

La quantité doit pouvoir être ajustée, et correctement prise en compte en temps réel dans le calcul du prix.

Le bouton pour supprimer un ourson du panier doit bien supprimer le bon ourson, la bonne ligne, autant dans les données du *local storage* que dans l'affichage en front.

Le formulaire doit etre correctement rempli, sans quoi la commande devient impossible.

Du code malveillant ne doit pas pouvoir être utilisé dans le formulaire.

Le bouton « passer votre commande » doit prendre en compte toutes les données de la page correctement, pour les transmettre à la page suivante.

Page confirmation de commande

L'API doit correctement renvoyer l'ID de la commande ainsi que le prix total. De plus, le panier doit être vidé, en prévision d'une prochaine commande (pour éviter de la redondance gênante).

Ceci peut être vérifié en vérifiant le contenu du *local storage* sur la page même, ou en retournant sur la page d'accueil pour préparer une autre commande (et constater le contenu du panier par la suite).