Plan unitaire

Projet 5

Openclassrooms

Damien Will

Sommaire

[1. Page index.JS 5](#_Toc80963370)

[1.1. Vérification de la bonne communication avec l’API : 5](#_Toc80963371)

[1.2. Si erreur de communication, identification de cette erreur : 5](#_Toc80963372)

[1.3. Vérification du type de réponse 5](#_Toc80963373)

[1.4. Vérification du fonctionnement des boutons permettant d’enregistrer le produit dans le localStorage : 5](#_Toc80963374)

[1.5. Vérification de l’enchainement des opérations liées aux promesses : 5](#_Toc80963375)

[2. Page produit.js 6](#_Toc80963376)

[2.1. Vérification du bon upload du produit stocké : 6](#_Toc80963377)

[2.2. Vérification fonction affichageProduit() 6](#_Toc80963378)

[2.3. Vérification fonction affichageValidation() 6](#_Toc80963379)

[2.4. Vérification fonction calculPrix() : 6](#_Toc80963380)

[2.5. Vérification liste d’options 6](#_Toc80963381)

[2.6. Validation des informations avant envoi panier 6](#_Toc80963382)

[2.7. Vérification des produits au panier : 7](#_Toc80963383)

[3. Page panier.js 7](#_Toc80963384)

[3.1. Vérification du bon upload des produits du panier : 7](#_Toc80963385)

[3.2. Vérification des regexs : 7](#_Toc80963386)

[3.3. Vérification fonction totalProduit() : 8](#_Toc80963387)

[3.4. Vérification panierVide() : 8](#_Toc80963388)

[3.5. Vérification du montant total de la commande 8](#_Toc80963389)

[3.6. Vérification du formulaire 8](#_Toc80963390)

[Vérification des champs vides : exemple pour le champ « prénom » : 8](#_Toc80963391)

[Vérification de la validité de tous les champs du formulaire : 8](#_Toc80963392)

[3.7. Vérification du tableau « panierFinal » qui sera envoyé à l’API : 9](#_Toc80963393)

[3.8. Vérification de la réussite de la requête POST : 9](#_Toc80963394)

[4. Page confirmation.js 9](#_Toc80963395)

[4.1. Vérification de la récupération des informations de la commande 9](#_Toc80963396)

[4.2. Vérification de la suppression du panier : 10](#_Toc80963397)

# Page index.JS

## Vérification de la bonne communication avec l’API :

*fetch("http://localhost:3000/api/cameras")*

*.then(response => {*

*if(response.ok){*

*console.log("response = OK")*

## Si erreur de communication, identification de cette erreur :

*else {*

*console.log("erreur : " + response.status);*

*}*

## Vérification du type de réponse

*.then(data => {*

*console.log(data);*

## Vérification du fonctionnement des boutons permettant d’enregistrer le produit dans le localStorage :

Sur Firefox, aller dans l’inspecteur, puis onglets « stockage »/ »stockage local »/http://127.0.0.1 :808.

Vérifier que la valeur de la clé « product » stockée  correspond bien au produit qui a été sélectionné par click au bouton.

## Vérification de l’enchainement des opérations liées aux promesses :

*.catch(function(err) {*

*console.log("err")*

*});*

# Page produit.js

## Vérification du bon upload du produit stocké :

*console.table(myProduct);*

## Vérification fonction affichageProduit()

Vérification visuelle

## Vérification fonction affichageValidation()

Vérification visuelle

## Vérification fonction calculPrix() :

*console.log(calculPrix(49900));//vérification de la fonction CalculPrix*

## Vérification liste d’options

Vérification visuelle

## Validation des informations avant envoi panier

*// Si aucune option n'est sélectionnée, message d'erreur et désactivation du bouton d'envoi*

*if(selectOption.selectedIndex == 0){*

*optionError.innerHTML = `! Merci de choisir une option`;*

*optionError.style.color = `red`;*

*event.preventDefault;*

*}*

*// Si aucune quantité n'est sélectionnée, message d'erreur et désactivation du bouton d'envoi*

*if(selectQuantite.selectedIndex == 0){*

*quantiteError.innerHTML = `! Merci de choisir une quantité`;*

*quantiteError.style.color = `red`;*

*event.preventDefault; }*

## Vérification des produits au panier :

Sur Firefox, aller dans l’inspecteur, puis onglets « stockage »/ »stockage local »/http://127.0.0.1 :808.

Vérifier que la valeur de la clé « panier » stockée correspond bien au(x) produit(s) sélectionné(s) par click au bouton.

# Page panier.js

## Vérification du bon upload des produits du panier :

*console.table(productsPanier);*

## Vérification des regexs :

*let regexLettres = /^[a-zA-Z-\s]+$/;*

*//test regex*

*if (regexLettres.test("dam27") == false){*

*console.log("regexLettres = OK");*

*}*

*else {console.log("regexLettres = NOT");}*

*let regexLettresChiffres = /^[a-zA-Z-\s0-9]+$/;*

*//test regex*

*if (regexLettresChiffres.test("123 dam") == true){*

*console.log("regexLettresChiffres = OK");*

*}*

*else {console.log("regexLettresChiffres = NOT");}*

*let regexEmail = /^[a-zA-Z0-9.!#$%&'\*+/=?^\_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)\*$/;*

*//test regex*

*if (regexEmail.test("dam27@com" && "dam27.com") == false){*

*console.log("regexEmail = OK");*

*}*

*else {console.log("regexEmail = NOT");}*

## Vérification fonction totalProduit() :

*let result1 = totalProduit(4,5);*

*if (result1 ===20){*

*console.log("fonction TotalProduit OK")*

*}*

*else {console.log("fonction TotalProduit NOT")}*

## Vérification panierVide() :

Vérification visuelle

## Vérification du montant total de la commande

Calcul mental sur les 3 1ères commandes

## Vérification du formulaire

Regexs déjà vérifiées.

### Vérification des champs vides : exemple pour le champ « prénom » :

*// Champ Prénom*

*if (firstName.value.trim() == "") {*

*errorPrenom.innerHTML = `Merci de remplir le champ "Prénom"`;*

*firstName.style.border= "1px solid red";*

*errorPrenom.style.color= "red";*

*event.preventDefault();*

*}*

Vérification visuelle.

### Vérification de la validité de tous les champs du formulaire :

*if (firstName.value.trim() != "" && regexLettres.test(firstName.value) != false && lastName.value.trim() != "" && regexLettres.test(lastName.value) != false && address.value.trim() != "" && regexLettresChiffres.test(address.value) != false && city.value.trim() != "" && regexLettres.test(city.value) != false && email.value.trim() != "" && regexEmail.test(email.value) != false){création d’un objet contact et d’un tableau de produits intégrés dans un tableau « panierFinal »}*

## Vérification du tableau « panierFinal » qui sera envoyé à l’API :

*//verif :*

*localStorage.setItem("order", JSON.stringify(panierFinal));*

Sur Firefox, aller dans l’inspecteur, puis onglets « stockage »/ »stockage local »/http://127.0.0.1 :808.

Vérifier que la valeur de la clé « order » stockée dans le localStorage correspond bien aux exigences requises par l’API pour la requête POST (pour cela, il faut désactiver : localStorage.clear(); de la page confirmation.js ou aller dans le débogueur de Firefox et mettre un point d’arrêt juste après la mise en stockage de panierFinal (ne pas oublier de remplir le formulaire !)).

## Vérification de la réussite de la requête POST :

*.then(requete =>{*

*if (requete.ok) {*

*console.log("requete = OK");*

*return requete.json();*

*} else {*

*console.log('Mauvaise réponse du serveur !')*

*}*

*})*

*.catch(error => {*

*console.log(error)*

*})*

# Page confirmation.js

## Vérification de la récupération des informations de la commande

console.log(commande);

console.log(montantTotal);

## Vérification de la suppression du panier :

Sur Firefox, aller dans l’inspecteur, puis onglets « stockage »/ »stockage local »/http://127.0.0.1 :808.

Vérifier qu’aucune clé n’est présente.