

Kanap



Contexte

- Je travaille dans une agence de développement
- Le client "Kanap", spécialisé dans la vente de canapés, souhaite mettre en place une plateforme e-commerce

Mission

À partir du travail front/back réalisé par mon équipe, je dois :

- Intégrer dynamiquement les éléments de l'API dans les pages web avec JavaScript
- Intégrer gestion du Panier / Formulaire de commande
- Réaliser un plan de test d'acceptation

Présentation

- 1 - Démonstration Live des fonctionnalités
- 2 - Présentation du code JavaScript
- 3 - Présentation du Plan de test d'acceptation

The screenshot shows a responsive e-commerce website for 'Kanap'. At the top, there's a header with a logo, contact info (01 23 45 67 89, support@name.com), and navigation links (Accueil, Panier). Below the header is a large image of a woman sitting cross-legged on a light-colored sofa, looking at her phone. This is followed by a section titled 'Nos produits' (Our products) with a subtitle 'Une gamme d'articles exclusifs'. A grid of eight sofa models is displayed, each with a small image, a name, and a short description. The footer contains the Kanap logo, address (10 quai de la charonne 75019 Paris 9), phone number (01 23 45 67 89), email (support@name.com), and a copyright notice (© Copyright 2021 - 2042 | Openclassrooms by Openclassrooms | All Rights Reserved | Powered by OpenClassrooms).

1 - Démonstration

2- *JavaScript*



JS *script.js*

1 - Récupérer les données de l'API

```
fetch("http://localhost:3000/api/products")
// Obtention des données de l'API => conversion du résultat en .json
.then((res) => res.json())
// Les données sont transmises sous forme de paramètre : "products" [...]
.then((products) => {
    // Affichage des données dans un tableau via la console
    console.table(products)
    // Appel de la fonction "hydrateProducts" + paramètre "products"
    hydrateProducts(products) Appel + (paramètre)
})
// Si ERREUR : Affichage via HTML + console
.catch((err) => {
    document.querySelector(".titles").innerHTML = "<h1>erreur 404</h1>"
    console.error("[API] erreur 404 : " + err) Erreur
});
```

```
[{"colors": [ 3 items ],  
 "_id": "107fb5b75607497b96722bda5b504926",  
 "name": "Kanap Sinopé",  
 "price": 1849,  
 "imageUrl": "http://localhost:3000/images/kanap01.jpeg",  
 "description": "Excepteur sint occaecat cupidatat non proident,  
 altTxt": "Photo d'un canapé bleu, deux places"},  
 { 7 items },  
 { 7 items }]
```

Test lien données API :
[api/products](#)

2 - Afficher les Produits

```
function hydrateProducts(products) {  
    // Déclaration + Pointage de la <section> ayant pour id "#items"  
    const productsArea = document.querySelector("#items") Cibler  
    // Pour chaque indice "product" de "products" [...]  
    for (const product of products) { Boucle sur données API  
        // Création de : <a href= "./product.html?id=${product._id}"> Ancre - URL page produit  
        productsArea.innerHTML += innerHTML  
        `<a href=". /product.html?id=${product._id}">  
        <article>  
              
            <h3 class="productName">${product.name}</h3>  
            <p class="productDescription">${product.description}</p>  
        </article>  
        </a>`  
    }  
}
```



JS

products.js

1 - Récupérer l'id produit dans l'URL

Barre d'adresse du navigateur :

```
① 127.0.0.1:5500/front/html/product.html?id=107fb5b75607497b96722bda5b504926
```

Méthode URLSearchParams :

```
const id = new URLSearchParams(window.location.search).get("id")
console.log("ID du produit à afficher :", id)
```

Contrôle du résultat dans la console :

ID du produit à afficher :
107fb5b75607497b96722bda5b504926

2 - Récupérer les données produit API

```
fetch(`http://localhost:3000/api/products/${id}`)
// Obtention des données de l'API => conversion du résultat en .json
.then((res) => res.json())
// Les données sont transmises sous forme de paramètre : "product" [...]
.then((product) => {
    // Appel de la fonction "hydrateProduct" + paramètre "product"
    hydrateProduct(product) Appel + (paramètre)
    console.log(product)
})
// Si ERREUR : Affichage via HTML + console
.catch((err) => {
    document.querySelector(".item").innerHTML = "<h1>erreur 404</h1>";
    console.error("API - erreur 404 : " + err) Erreur
})
```

Contrôle du résultat dans la console :

Données de Kanap Sinopé récupérées : product.js:22

```
{colors: Array(3), _id: '107fb5b75607497b96722bda5b504926',
▼ name: 'Kanap Sinopé', price: 1849, imageUrl: 'http://Localho
st:3000/images/kanap01.jpeg', ...} ⓘ
  altTxt: "Photo d'un canapé bleu, deux places"
▶ colors: (3) ['Blue', 'White', 'Black']
  description: "Excepteur sint occaecat cupidatat non proident
  imageUrl: "http://localhost:3000/images/kanap01.jpeg"
  name: "Kanap Sinopé"
  price: 1849
  _id: "107fb5b75607497b96722bda5b504926"
```

3 - Afficher le produit

```

function hydrateProduct(product) {
    // Titre de la page : Affichage dans l'onglet de navigation
    Titre page document.title = `${product.name} | Kanap`

    // Nom, Prix, Description
    Nom document.querySelector('#title').textContent = product.name textContent
    Prix document.querySelector('#price').textContent = product.price
    Description document.querySelector('#description').textContent = product.description

    // Image + Alt
    const imageParent = document.querySelector(".item__img")
    const image = document.createElement("img") createElement
    image.src = product.imageUrl
    image.alt = product.altTxt
    imageParent.appendChild(image) appendChild

    // Éléments : Couleurs disponibles
    const colorsParent = document.querySelector("#colors") Cibler
    const colors = product.colors
    colors.forEach((color) => { Boucle données color du produit
        const colorSettings = document.createElement("option")
        colorSettings.value = color
        colorSettings.textContent = color
        colorsParent.appendChild(colorSettings)
    })
    // Appel de la fonction + transmission du paramètre "product"
    productToPurchase(product) Appel + (paramètre)
}

```



Kanap Sinopé

Prix : 1849€

Description :

Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Choisir une couleur :

Nombre d'article(s) (1-100) :

[Ajouter au panier](#)

4 - Créer un objet à ajouter au Panier

Création d'un objet "Achat"

```
function productToPurchase(product) {
    // Écoute de l'élément #addToCart via méthode "addEventListener"
    const buttonAddToCart = document.querySelector('#addToCart') Cibler
    buttonAddToCart.addEventListener("click", () => { Écouter "clic" du bouton
        // Récupération des valeurs input de #colors & #quantity
        const color = document.querySelector('#colors').value
        const quantity = document.querySelector('#quantity').value
        // Création de la classe qui sera utilisée pour ajouter au panier
        const purchase = {
            id: product._id,
            color: color,
            quantity: Number(quantity),
            name: product.name
        }
        console.log(`Préparation de ${purchase.name}:`, purchase)
    })
}

Appels : // Contrôle de la validité de l'achat (Couleur / Quantité)
if (purchaseInvalid(purchase, color, quantity)) return
// Ajout du produit au Panier du Local Storage
addToCart(purchase, color)
// Réinitialisation du bouton "Ajouter au panier" après un achat
resetButton()
}
```

Au clic du bouton :

Ajouter au panier

Const purchase =

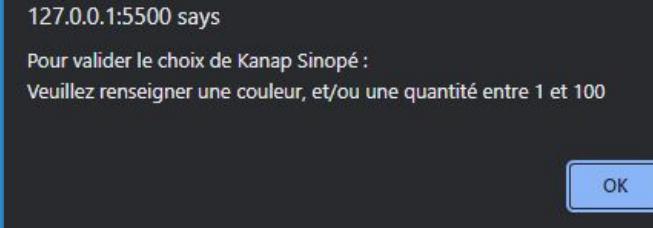
```
Préparation de Kanap Sinopé: product.js:80
  {id: '107fb5b75607497b96722bda5b504926', color: 'Blue', quantity: 1, name: 'Kanap Sinopé'} ⓘ
    color: "Blue"
    id: "107fb5b75607497b96722bda5b504926"
    name: "Kanap Sinopé"
    quantity: 1
  ► [[Prototype]]: Object
```

5 - Contrôler la validité de l'ajout

Fonction :

```
function purchaseInvalid(purchase, color, quantity) {
    // Invalide si : Couleur non définie / Quantité inférieure à 1 ou supérieure à 100 [...]
    if (!color || quantity < 1 || quantity > 100) {
        console.error(`Erreur lors de l'ajout au Panier de ${purchase.name} ! \n Couleur et/ou Quantité invalides :`, purchase)
        alert(`Pour valider le choix de ${purchase.name} : \nVeuillez renseigner une couleur, et/ou une quantité entre 1 et 100`)
        return true  Erreur
    }
}
```

Fenêtre d'alerte :



Résultat console :

```
✖ ▶ Erreur lors de l'ajout au Panier de Kanap Sinopé !          product.js:97
  Couleur et/ou Quantité invalides :
  ▼ {id: '107fb5b75607497b96722bda5b504926', color: '', quantity: 0, name: 'Kanap Sinopé'}
    color: ""
    id: "107fb5b75607497b96722bda5b504926"
    name: "Kanap Sinopé"
    quantity: 0
  ▶ [[Prototype]]: Object
```

6 - Utiliser le Local Storage (Panier)

Ajouter 1er produit & Créer Panier :

```
function addToCart(purchase, color) {
    // Déclaration du Panier, clé "Cart" dans Local Storage
    let myCart = JSON.parse(localStorage.getItem("Cart")) Déclarer Panier

    // Si "myCart" est vide -> Création d'un tableau -> Ajout du 1er produit
    if (myCart == null) { Si Panier vide ...
        myCart = []
        myCart.push(purchase)
        localStorage.setItem("Cart", JSON.stringify(myCart))
        // Confirmation de l'ajout au panier Envoyer 1er objet dans un
        purchaseConfirmation(purchase) tableau appelé Cart
    }

    // Ajouter d'autres produits : 1er cas de figure ->
    // Si "myCart" n'est pas vide [...]
    else if (myCart != null) { ...
    }
}
```

Sinon ... ->

Observer les résultats dans le Local Storage :

```
[{"id": "107fb5b75607497b96722bda5b504926", "color": "Blue", "quantity": 10, "name": "Kanap Sinopé"}, ...
0: {"id": "107fb5b75607497b96722bda5b504926", "color": "Blue", "quantity": 10, "name": "Kanap Sinopé"}
1: {"id": "107fb5b75607497b96722bda5b504926", "color": "White", "quantity": 1, "name": "Kanap Sinopé"}
2: {"id": "107fb5b75607497b96722bda5b504926", "color": "Black", "quantity": 1, "name": "Kanap Sinopé"}
3: {"id": "415b7cacb65d43b2b5c1ff70f3393ad1", "color": "Black/Yellow", "quantity": 5, "name": "Kanap Cyllène"}
4: {"id": "415b7cacb65d43b2b5c1ff70f3393ad1", "color": "Black/Red", "quantity": 1, "name": "Kanap Cyllène"}
```

Ajouter d'autres produits :

```
// Ajouter d'autres produits : 1er cas de figure ->
// Si "myCart" n'est pas vide [...]
else if (myCart != null) {
    // Boucle sur les éléments de "myCart"
    for (i = 0; i < myCart.length; i++) { Boucler Panier
        if (
            // Si le produit à ajouter est similaire : id/couleur
            myCart[i].id == purchase.id && myCart[i].color == color
        ) {
            return (
                // => Ajout de quantité au produit déjà dans le panier
                myCart[i].quantity = Math.min(myCart[i].quantity + purchase.quantity, 100),
                localStorage.setItem("Cart", JSON.stringify(myCart)),
                // Confirmation de l'ajout au panier
                purchaseConfirmation(purchase)
            )
        }
    }
}

// Ajouter d'autres produits : 2eme cas de figure ->
// Boucle sur les éléments de "myCart"
for (i = 0; i < myCart.length; i++) { Boucler Panier
    // Si produit à ajouter = ID similaire et couleur différente (OU) ID différent
    if (myCart[i].id == purchase.id && myCart[i].color != color || myCart[i].id != purchase.id) id identique mais couleur différente
        (ou) id différent
    ) {
        return (
            // => Ajout d'un nouvel article dans le panier
            myCart.push(purchase),
            localStorage.setItem("Cart", JSON.stringify(myCart)),
            // Confirmation de l'ajout au panier
            purchaseConfirmation(purchase)
        )
    }
}
```

a - Ajouter quantité à produit identique

Ajout quantité

b- Ajouter un nouveau produit

Ajout nv produit

7 - Confirmer l'ajout de l'article au Panier

Fonction :

```
function purchaseConfirmation(purchase) {
    // Confirmations dans la Console
    console.log(`${purchase.name} ${purchase.color} ajouté au Panier`, purchase)
    let myCart = JSON.parse(localStorage.getItem("Cart"))
    console.log("Panier à jour :", myCart)

    // Fenêtre de confirmation dans le navigateur
    if (window.confirm(`${purchase.name} option: ${purchase.color} a bien été ajouté au panier !
        Consulter le Panier [OK] | Rester à ${purchase.name} [Annuler]`)) {
        window.location.href = "cart.html"
    } else {
        window.close
    }
    // Changement du style visuel du bouton "Ajouter au panier" (couleur/texte)
    document.querySelector("#addToCart").style.color = "rgb(0, 205, 0)"
    document.querySelector("#addToCart").textContent = "Produit ajouté !"  Style Bouton
}
```

Résultat console

Alerte navigateur

Style Bouton

Contrôle des résultats dans la console :

```
Kanap Cyllène Black/Yellow ajouté au Panier : product.js:164
{id: '415b7cacb65d43b2b5c1ff70f3393ad1', color: 'Black/Yellow',
 quantity: 1, name: 'Kanap Cyllène'}
```

Alerte navigateur :

127.0.0.1:5500 says

Kanap Cyllène option: Black/Yellow a bien été ajouté au panier !
 Consulter le Panier [OK] | Rester à Kanap Cyllène [Annuler]

OK

Cancel

Style Bouton :

Produit ajouté !

 ***cart.js***

1 - Récupérer les données non stockés dans le Local Storage

a) Récupérer les données de l'API :

```
fetch("http://localhost:3000/api/products")
  // Obtention des données de l'API => conversion du résultat en .json
  .then((res) => res.json())
  // Les données sont transmises sous forme de paramètre : "products" [...]
  .then((products) => {
    console.log("API :", products)
    // Appel de la fonction "getProducts" + paramètre "products"
    getPurchaseData(products) Appel + (paramètre)
  })
  // Si ERREUR : Affichage via HTML + console
  .catch((err) => {
    document.querySelector("#cartAndFormContainer").innerHTML = "<h1>erreur 404</h1>"
    console.log("API - erreur 404 : " + err) Error
  })
}
```

b) Ajouter les données aux produits du Panier :

```
function getPurchaseData(products) {
  // Récupération de notre panier "Cart"
  const myCart = JSON.parse(localStorage.getItem("Cart"));
  // Si mon panier n'est pas vide [...]
  if (myCart != null) {
    // Boucle sur les produits du panier
    for (let purchase of myCart) {
      // Boucle sur les produits de l'API
      for (let a = 0, b = products.length; a < b; a++) {
        // Si id produit PANIER = id produit API
        if (purchase.id === products[a]._id) {
          // Récupération des informations manquantes
          purchase.name = products[a].name;
          purchase.price = products[a].price;
          purchase.imageUrl = products[a].imageUrl;
          purchase.altTxt = products[a].altTxt;
          purchase.description = products[a].description;
        }
      }
    }
    // Affichage console :
    // "myCart" possède les valeurs du Local Storage + celles récupérées au dessus
    // Les valeurs récupérées ne sont pas envoyées au Local Storage
    console.log("Panier :", myCart)
    // Appel de la fonction "hydrateCart" + paramètre "myCart" qui cumule les valeurs
    hydrateCart(myCart); Appel affichage + (paramètre)
  } else {
    // Si le Panier est vide :
    document.querySelector("#totalQuantity").innerHTML = "0";
    document.querySelector("#totalPrice").innerHTML = "0";
    document.querySelector("h1").innerHTML =
      "Vous n'avez pas d'article dans votre panier";
  }
}
```

Boucle produits panier

Boucle produits API

Si correspondance id ...

Si panier vide

c) Contrôler les résultats = Les données sont ajoutées

```
Panier :                                         cart.js:55
▼ (5) [{} , {} , {} , {} , {}] ⓘ
  ▼ 0:
    altTxt: "Photo d'un canapé bleu, deux places"
    color: "Blue"
    description: "Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."
    id: "107fb5b75607497b96722bda5b504926"
    imageUrl: "http://localhost:3000/images/kanap01.jpeg"
    name: "Kanap Sinopé"
    price: 1849
    quantity: 10
```

2 - Afficher le contenu du Panier

```
function hydrateCart(myCart) {
    // Déclaration + Pointage de la zone d'affichage du Panier
    const cartArea = document.querySelector("#cart_items"); Cibler id "cart_items"
    // Création du HTML dynamique : Méthode .map() + introduction de data-set
    cartArea.innerHTML += myCart.map((purchase) =>
        `<article class="cart_item" data-id="${purchase.id}" data-color="${purchase.color}" data-quantity="${purchase.quantity}" data-price="${purchase.price}">
            <div class="cart_item_img">
                
            </div>
            <div class="cart_item_content">
                <div class="cart_item_content_titlePrice">
                    <h2>${purchase.name}</h2>
                    <span>Couleur : ${purchase.color}</span>
                    <p data-price="${purchase.price}">Prix : ${purchase.price} €</p>
                </div>
                <div class="cart_item_content_settings">
                    <div class="cart_item_content_settings_quantity">
                        <p>Quantité : </p>
                        <input type="number" class="itemQuantity" name="itemQuantity" min="1" max="100" value="${purchase.quantity}">
                    </div>
                    <div class="cart_item_content_settings_delete">
                        <p class="deleteItem" data-id="${purchase.id}" data-color="${purchase.color}">Supprimer</p>
                    </div>
                </div>
            </div>
        </article>
    ).join("")
    // ".join()" permet de définir la jonction entre chaque <article> affiché
    // Par défaut c'est une virgule, on la remplace par un espace vide
}

// Appel des fonctions pour écoute :
updateQuantity() // Modification de la quantité
deletePurchase() // Suppression d'un produit
totalCart() // Calcul total produits/prix panier
}
```

Utiliser dataset

innerHTML + méthode MAP
= itérer fonction sur chaque article Panier

Appels : modifier quantité, supprimer, calcul total

The screenshot shows a user interface for a shopping cart. It displays two items:

- Kanap Sinopé**: A blue sofa. Details: Couleur : Blue, Prix : 1849 €. Quantity input field shows 10, with a 'Supprimer' button.
- Kanap Cyllène**: A yellow sofa. Details: Couleur : Black/Yellow, Prix : 4499 €. Quantity input field shows 7, with a 'Supprimer' button.

3 - Afficher les totaux quantité & prix

```
function totalCart() {  
    // Déclaration des variables de "Total" en tant que Number  
    let totalProducts = 0    Déclarer variables résultats totaux  
    let totalPrice = 0  
    // Déclaration + Pointage de tous les éléments ".cart_item"  
    const purchases = document.querySelectorAll(".cart_item") Cibler  
    // Boucle : pour chaque élément "purchase" des purchases  
    purchases.forEach((purchase) => { Boucle ".cart_item" + Utiliser dataset : quantity & price  
        // Récupération des quantités des produits via les dataset  
        (Récupérer) totalProducts += JSON.parse(purchase.dataset.quantity)  
        // Calcul de prix panier total via les dataset  
        (Calcul) totalPrice += purchase.dataset.quantity * purchase.dataset.price  
    });  
    // Affichage des résultats dans le HTML  Affichage  
    document.getElementById("totalQuantity").textContent = totalProducts  
    document.getElementById("totalPrice").textContent = totalPrice  
}
```



Kanap Sinopé
Couleur : Blue
Prix : 1849 €

Quantité : Supprimer



Kanap Cyllène
Couleur : Black/Yellow
Prix : 4499 €

Quantité : Supprimer

Total (6 articles) : 19044 €

Affichage

4 - Modifier la Quantité d'un article

```
function updateQuantity() {
    // Déclaration + Pointage de tous les éléments ".cart__item"
    const cartArea = document.querySelectorAll(".cart__item"); Cibler

    // Écoute des évènements (eQuantity) sur chaque article ("itemQuantity")
    cartArea.forEach((purchase) => { Boucle ".cart__item"
        purchase.addEventListener("change", (eQuantity) => { Écouter quantité
            const myCart = JSON.parse(localStorage.getItem("Cart"))
            // Boucle sur les produits du Panier
            for (product of myCart) Boucle Panier
                if (
                    // Si id + color similaires
                    product.id === purchase.dataset.id && Si correspondance id +
                    purchase.dataset.color === product.color couleur ...
                ) {
                    // Modification de la quantité
                    product.quantity = Math.min(eQuantity.target.value, 100)
                    // Envoi au Local Storage
                    localStorage.Cart = JSON.stringify(myCart) Mise à jour Panier
                    // Mise à jour du dataset quantity
                    purchase.dataset.quantity = eQuantity.target.value
                    // Appel de la fonction de Total dynamique
                    totalCart(); Appel calcul totaux Panier
                }
                console.log("Article modifié :", product.name, purchase.dataset.color)
                console.log("Panier mis à jour :", myCart)
            }
        })
    })
}
```

".cart__item" =



5 - Supprimer un article du Panier

```
function deletePurchase() {
    // Déclaration + Pointage de tous les éléments ".cart_item .deleteItem"
    const deletePurchase = document.querySelectorAll(".cart_item .deleteItem") Cibler boutons "Supprimer"
    // Pour chaque éléments [...]
    deletePurchase.forEach((purchase) => { Boucle ".cart_item .deleteItem"
        // Écoute du click sur bouton "Supprimer" de chaque produit
        purchase.addEventListener("click", () => [ Écouter clic sur chaque bouton (usage dataset id /color)
            // Appel du Panier en Local Storage
            let myCart = JSON.parse(localStorage.getItem("Cart"))
            // Boucle : Pour chaque élément du Panier [...]
            for (let a = 0, b = myCart.length; a < b; a++) Boucle Panier
            if (
                // Si correspondance Panier/dataset (id/color)
                myCart[a].id === purchase.dataset.id &&
                myCart[a].color === purchase.dataset.color
            ) {
                // Variable utile pour suppression
                const picked = [a];
                // Suppression de 1 élément à l'indice num "splice" Panier = Suppression article
                myCart.splice(picked, 1)
                // Renvoi du (nouveau) panier converti dans le Local Storage
                localStorage.Cart = JSON.stringify(myCart) Envoyer nouveau Panier

                // Suppression de l'Affichage HTML
                const displayToDelete = document.querySelector( Supprimer article de l'affichage
                    `article[data-id="${purchase.dataset.id}"][data-color="${purchase.dataset.color}"]`)
                displayToDelete.remove()

                totalCart(); // Appel de la fonction Total Quantité/Prix
            } Appel calcul totaux Panier
        })
    })
}
```

".cart_item .deleteItem" =



Kanap Cyllène
Couleur : Black/Yellow
Prix : 4499 €
Quantité : 10
Supprimer

Contrôle résultat via la console :

```
Panier mis à jour : cart.js:135
▼ (2) [{...}, {...}] ⓘ
  ► 0: {id: '107fb5b75607497b96722bda5b504926'}
  ► 1: {id: '415b7cacb65d43b2b5c1ff70f3393ad1'
    length: 2
  [[Prototype]]: Array(0)
Article supprimé cart.js:193
Panier mis à jour : cart.js:194
▼ [...] ⓘ
  ► 0: {id: '107fb5b75607497b96722bda5b504926'
    length: 1
  [[Prototype]]: Array(0)
```

6 - Écouter le Formulaire

```
function listenToForm() {  
    // Formulaire : Bouton "Commander"  
    const orderButton = document.querySelector("#order")  
    // => Appel de la fonction "submitForm" Écouter clic  
    orderButton.addEventListener("click", (e) => submitForm(e))  
    // Déclaration + Pointage de l'élément  
    let firstNameField = document.querySelector('#firstName')  
    // Écoute d'évènement au niveau de l'input  
    firstNameField.addEventListener('input', checkFirstName)  
    // Cibler  
    let lastNameField = document.querySelector('#lastName')  
    lastNameField.addEventListener('input', checkLastName)  
    // Ecouter input  
    let cityField = document.querySelector('#city')  
    cityField.addEventListener('input', checkCity)  
  
    let addressField = document.querySelector('#address')  
    addressField.addEventListener('input', checkAddress)  
  
    let emailField = document.querySelector('#email')  
    emailField.addEventListener('input', checkEmail)  
}
```

Bouton Commander
Écouter clic
Appel "Envoyer"

Cibler
Ecouter input
Appel

Exemple : Fonction contrôle Email

```
function checkEmail() {  
    let emailInput = document.getElementById("email")  
    let emailValidate = document.getElementById("email").value  
    let emailErrorMsg = document.getElementById("emailErrorMsg")  
  
    const emailRGEX = /^[\\w-.]+@[\\w-\\.]+[\\w-]{2,4}$/ REGEX  
    let emailResult = emailRGEX.test(emailValidate)  
  
    if (emailResult == false) {  
        emailInput.style.backgroundColor = "red"  
        emailInput.style.color = "white"  
        emailErrorMsg.innerHTML = `Champ requis<br>Exemple : moi@kanap.com`  
        emailErrorMsg.style.display = "inherit"  
        return false Échec  
    } else {  
        emailInput.style.backgroundColor = "rgb(0, 205, 0)"  
        emailInput.style.color = "black"  
        emailErrorMsg.style.display = "none"  
        return true Succès  
    }  
}
```

Adresse:
25 rue Guillaume Lekeu

Ville:
Angers4

Champ requis:
- "Ville" ne doit comporter que des lettres
- Tires, apostrophes, et accents sont autorisés

Email:
Franck.com

Champ requis
Exemple : moi@kanap.com

Affichage :

7 - Valider le Formulaire

```
function buildForm(e) {  
    // Déclaration et pointage des éléments nécessaires  
    const myCart = JSON.parse(localStorage.getItem("Cart"))  
    const firstName = document.getElementById("firstName").value  
    const lastName = document.getElementById("lastName").value  
    const address = document.getElementById("address").value  
    const city = document.getElementById("city").value  
    const email = document.getElementById("email").value  
    // Constante : Appel des fonctions de validation  
    const formValid = checkEmail() && checkAddress() && checkCity() && checkFirstName() && checkLastName()  
  
    Conditions nécessaires à la validation finale du formulaire  
    if (myCart !== null && [firstName, lastName, address, city, email] !== '' && formValid) {  
  
        // Récupération des id(s) Produits du Panier  
        const productsIds = []  
        myCart.forEach((purchase) => {  
            productsIds.push(purchase.id)  
        })  
  
        const form = {  
            // Objet respectant les attentes de l'API  
            contact: {  
                firstName: firstName,  
                lastName: lastName,  
                address: address,  
                city: city,  
                email: email  
            },  
            products: productsIds  
        }  
        return form  
  
    } else {  
        console.error("Champs invalides et/ou Panier vide")  
        alert("Formulaire invalide et/ou Panier vide.\nNote : TOUS les champs sont requis !")  
        e.preventDefault()  
    }  
}
```

Récupération des entrées Formulaire

Récupérer id(s) produit(s)

Construire objet pour Fetch POST

ERREUR

Fenêtre d'alerte navigateur si erreur :

127.0.0.1:5500 says
Formulaire invalide et/ou Panier vide.
Note : TOUS les champs sont requis !

OK

8 - Envoyer le Formulaire / Obtenir l'id de commande

```
function submitForm(e) {
    // Récupération du Formulaire valide
    const form = buildForm(e) Récupérer "form" par appel de fonction "buildForm"
    // Si Formulaire invalide : Envoi annulé
    if (form == null) return

    fetch("http://localhost:3000/api/products/order", {
        method: "POST",
        headers: {
            'Accept': 'application/json',
            "Content-Type": "application/json",
        },
        body: JSON.stringify(form), "Stringifier" le body à envoyer
    })
        .then(res => res.json())
        .then(res => {
            console.log("Formulaire de commande : ", res) Succès !
            alert("Votre commande a bien été effectuée !")
            window.location.replace(`./confirmation.html?orderId=${res.orderId}`)
        })
        .catch(err) => {
            alert(err.message) Erreur
            console.log(err)
        }
}
```

Contrôle des résultats via la console :

```
Formulaire de commande : cart.js:259
{contact: {...}, products: Array(1), orderId: 'e5891a6
-0f63-11ed-af60-8d44692a2278'} i
  ▼ contact:
    address: "25 rue Guillaume Lekeu"
    city: "Angers"
    email: "franck.vukelic@gmail.com"
    firstName: "Franck"
    lastName: "VUKELIC"
    ► [[Prototype]]: Object
    orderId: "e5891a60-0f63-11ed-af60-8d44692a2278"
  ▼ products: Array(1)
    ▶ 0: {colors: Array(3), _id: '107fb5b75607497b967221
      length: 1
```

Fenêtre d'alerte navigateur :

127.0.0.1:5500 says
Votre commande a bien été effectuée !

OK

JS confirmation.js

```
/*
 ** MAIN | Variables / Constantes / Appels de Fonctions
 /**
 * Récupération du Numéro de Commande          URLSearchParams
 */
function getOrderId() {
  const urlParams = new URLSearchParams(location.search)
  return urlParams.get("orderId")
// return new URL(window.location.href).searchParams.get('orderId')
}

/**
 * Affichage du Numéro de Commande          Affichage du order id
 */
function displayOrderId(orderId) {
  const orderIdElement = document.getElementById("orderId")
  orderIdElement.textContent = orderId
}

/**
 * Suppression du Panier          Vider Local Storage
 */
function cleanCache() {
  const cache = window.localStorage
  cache.clear()
}
```

Barre d'adresse du navigateur :



Affichage du numéro de commande :



Panier remis à zéro :

You n'avez pas d'article dans votre panier

Total (0 articles) : 0 €

3 - *Plan de Test*