# PROJET KANAP


Site e-commerce

Javascript

# La page d'Accueil - Objectifs

- Elle doit présenter les 8 articles de l'API

- Elle doit afficher le nom, l'image et le début de la description de chaque article

# La page d'Accueil - CODE

1 _ On cible la section

```
//On cible la section
const section = document.getElementById("items");
```

2 _ On utilise FETCH pour récupérer les données de l'API

```
// On récupère les données avec Fetch
fetch("http://localhost:3000/api/products")
    .then((response) => response.json())
    .then((products) => {
```

3_ On crée une boucle "for"

```
.then((products) => {
  for (let i = 0; i < products.length; i++) {
```

# 4_ On crée les détails du produit à afficher

```
    // création du lien, élément enfant de la section
    let lien = document.createElement("a");
    lien.href = `./product.html?id=${products[i]._id}`;
    section.append(lien);
    // création de l'article, élément enfant du lien
    let article = document.createElement("article");
    lien.append(article);
    // création des éléments enfants de l'article
    // image
    let productImage = document.createElement("img");
    productImage.src = products[i].imageUrl;
    productImage.alt = products[i].altTxt;
    article.append(productImage);
    //nom
    let productNameH3 = document.createElement("h3");
    productNameH3.classList.add("productName");
    productNameH3.textContent = products[i].name;
    article.append(productNameH3);
    //description
    let productDescriptionP = document.createElement("p");
    productDescriptionP.classList.add("productDescription");
    productDescriptionP.textContent = products[i].description;
    article.append(productDescriptionP);
}
```

# La page Produit - Objectifs

- Elle doit présenter un seul produit, avec ses détails (images, prix, description, nom)

- Elle doit avoir un menu déroulant pour les couleurs et un input pour les quantités

- Ces éléments doivent être pris en compte lors de l'ajout du produit au panier

# La page Produit - CODE

➢Trois grandes parties : la redirection, l'affichage et l'ajout au panier

**1 _ Redirection: On isole l'id de l'article cliqué avec URLSearchParams**

```
let params = new URL(document.location).searchParams;
let productId = params.get("id");
```

**2 _ Affichage: On utilise Fetch pour récupérer les détails du produit à afficher**

```
fetch(`http://localhost:3000/api/products/${productId}`)
  .then((response) => response.json())
  .then((product) => {
```

**3 _ Affichage: On récupère de nouveau les détails du produit (image, nom, prix, description)**

## 4 _ Affichage: L'affichage des options de couleur grâce à une boucle "for"

```
let selectOption = document.querySelector("select");
for (i = 0; i < product.colors.length; i++) {
  let option = document.createElement("option");
  option.setAttribute("value", product.colors[i]);
  option.textContent = product.colors[i];
  selectOption.append(option);
}
```

## 5 _ Ajout : L'addEventListener

```
let add = document.getElementById("addToCart");
add.addEventListener("click", function () {
```

## 6 _ Ajout: Contrôle des couleurs et quantité

```
let colorValue = document.getElementById("colors").value;
if (colorValue == "") {
  return alert("Veuillez choisir une couleur");
}
let quantityValue = parseInt(document.getElementById("quantity").value);
if (quantityValue < 1 || quantityValue > 100) {
  return alert("Veuillez choisir une quantité entre 1 et 100");
}
```

## 7 _ Ajout : On définit l'objet article, qui sera stocké dans le LocalStorage

```
let article = {
    id: productId,
    color: colorValue,
    quantity: quantityValue,
};
```

## 8 _ Ajout : On vérifie si le panier est vide, et les actions qui en découlent

```
if (getPanier == null) {
  //on crée le tableau de produit et on ajoute le premier produit
  let panier = [];
  panier.push(article);
  localStorage.setItem("panier", JSON.stringify(panier));
} else {
```

## 8 _ Ajout : On vérifie si le panier n'est pas vide, et les actions qui en découlent

```
} else {
  // si le panier est déja existant
  let storage = JSON.parse(getPanier);
  // on cherche un produit avec mm couleur & id
  let foundProduct = storage.find(
    (p) => p.id == article.id && p.color == article.color
  );
  if (foundProduct != undefined) {
    // si on trouve un produit idtq on met a jour la qté
    foundProduct.quantity += article.quantity;
    localStorage.setItem("panier", JSON.stringify(storage));
  } else {
    //sinon on ajoute un nouveau produit
    storage.push(article);
    localStorage.setItem("panier", JSON.stringify(storage));
  }
}
```

# La page Panier - Objectifs

- Elle doit afficher les produits qui ont été stockés dans le LocalStorage lors de "l'ajout au panier", ainsi que le prix et la quantité total

- L'utilisateur doit avoir la possibilité de modifier ou de supprimer un produit du panier, avec un ajustement des totaux

- Les champs du formulaire doivent être analysés et validés pour vérifier le format et le type de données avant l'envoi à l'API

# La page Panier - Objectifs

- En cas d'absence ou d'erreur de saisie, un message d'erreur doit apparaitre sous le champ concerné

- Pour l'envoi au serveur, le panier doit contenir au moins un article et le formulaire doit être valide. Sinon, une alerte doit apparaitre sur la page.

# La page Panier - CODE

➢ Deux grandes parties : le panier et le formulaire

**1 _ On récupère le panier depuis le LocalStorage**

```
let produitDuPanier = JSON.parse(localStorage.getItem("panier"));
```

**2 _ On vérifie, la présence d'article dans le panier, s'il n'y en a pas l'utilisateur reçoit une alerte**

```
// on cible l'emplacement
const cartItem = document.getElementById("cart__items");
//si le panier ne contient aucun article
if (produitDuPanier === null) {
  // indiquer que le panier est vide
  alert("Votre panier est vide, veuillez séléctionner au moins un article");
```

**3 _ S'il y en a, on les affiche avec le prix et la quantité total. Pour cela on crée les tableaux de prix et quantité total**

```
//on définis les variables
let totalPrice = [];
let totalQuantity = [];
```

**4 _ On crée la boucle "for" pour afficher chaque produit du panier, on définit à l'intérieur les variables pour l'id, la couleur et la quantité des articles**

```
for (let i = 0; i < produitDuPanier.length; i++) {
  //On isole L'id du produit + couleur et qty
  let productId = produitDuPanier[i].id;
  let productColor = produitDuPanier[i].color;
  let productQuantity = produitDuPanier[i].quantity;
```

**5 _ On utilise de nouveau FETCH pour récupérer les produits du panier que l'on affiche cette fois avec innerHTML**

```
fetch(`http://localhost:3000/api/products/${productId}`)
  .then((response) => response.json())
  .then((product) => {
    // console.log(product, produitDuPanier[i]);
    cartItem.innerHTML += `                <article class="cart__item" data-id="${productId}" data-color="${prod
    <div class="cart__item__img">
      <img src="${product.imageUrl}" alt="${product.altTxt}">
    </div>
    <div class="cart__item__content">
      <div class="cart__item__content__description">
        <h2>${product.name}</h2>
        <p>${productColor}</p>
        <p>${product.price}</p>
      </div>
      <div class="cart__item__content__settings">
        <div class="cart__item__content__settings__quantity">
          <p>Qté : </p>
          <input type="number" class="itemQuantity" name="itemQuantity" min="1" max="100" value="${productQuant
        </div>
        <div class="cart__item__content__settings__delete">
          <p class="deleteItem">Supprimer</p>
        </div>
      </div>
    </div>
  </article>`;
```

## 6 _ On calcule la quantité total, puis le prix total grâce notamment à la méthode "reducer", puis nous les affichons dynamiquement

```
// quantity total
totalQuantity.push(parseInt(productQuantity));
console.log(totalQuantity);

// prix total
let productPrice = product.price;
let sousTotal = productPrice * productQuantity;
totalPrice.push(sousTotal);
console.log(totalPrice);

// on fait le calul grace a la fonction reducer
let reducer = (accumulator, currentValue) => accumulator + currentValue;
let TotalQuantityPanier = totalQuantity.reduce(reducer, 0);
let TotalPricePanier = totalPrice.reduce(reducer, 0);
console.log(TotalQuantityPanier);
console.log(TotalPricePanier);

// on affiche le total et le nombre d'article
let quantity = document.getElementById("totalQuantity");
quantity.textContent = ` ${TotalQuantityPanier}`;
let total = document.getElementById("totalPrice");
total.textContent = `${TotalPricePanier}`;
```

## 7 _ Nous créons une nouvelle boucle, mais sur l'input quantité, afin de gérer la modification dans le panier

```
let itemQuantity = document.querySelectorAll(".itemQuantity");
console.log(itemQuantity);

for (let i = 0; i < itemQuantity.length; i++) {
```

## 8 _ La méthode Element.closest() pour gérer le changement de quantité

```
itemQuantity[i].addEventListener("change", (e) => {
    let newQuantity = e.target.value;
    let articleToChangeId = e.target.closest("article").dataset.id;
    let articleToChangeColor =
      e.target.closest("article").dataset.color;

    let foundProduct = produitDuPanier.find(
      (p) =>
        p.id === articleToChangeId && p.color === articleToChangeColor
    );
    if (foundProduct != undefined) {
      foundProduct.quantity = newQuantity;
      localStorage.setItem("panier", JSON.stringify(produitDuPanier));
      window.location.reload();
```

## 9 _ La suppression d'un article

```
//Suppresion d'un article
let btnSupprimer = document.querySelectorAll(".deleteItem");
for (let i = 0; i < btnSupprimer.length; i++) {
  btnSupprimer[i].addEventListener("click", (e) => {
    let articleToDeleteId = e.target.closest("article").dataset.id;
    let articleToDeleteColor =
      e.target.closest("article").dataset.color;
    let foundProduct = produitDuPanier.find(
      (p) =>
        p.id === articleToDeleteId && p.color === articleToDeleteColor
    );
    if (foundProduct != undefined) {
      produitDuPanier.splice(i, 1);
      localStorage.setItem("panier", JSON.stringify(produitDuPanier));
      window.location.reload();
    }
```

```
// Lorsque tout les articles ont été supprimés du panier
if (produitDuPanier.length === 0) {
  totalQuantity = document.getElementById("totalQuantity");
  totalQuantity.textContent = 0;
  totalPrice = document.getElementById("totalPrice");
  totalPrice.textContent = 0;
  alert("Le panier a été vidé, veuillez sélectionner au moins un article");
}
```

## 10 _ Validation formulaire : ciblage

```
let validation = document.getElementById("order");
```

## 11 _ Vérification des champs: les variables

```
let prenom = document.getElementById("firstName");
let prenomErreur = document.getElementById("firstNameErrorMsg");
let prenomRegexp = /^(([A-Za-zÉÈÎÏéèêîïàç]+['.]?[ ]?|[a-zéèêîïàç]+['-]?)+)$/;
validation.addEventListener("click", validatePrenom);
```

## 12 _ Vérification des champs: les fonctions

```
function validatePrenom(e) {
    e.preventDefault();
    if (prenom.validity.valueMissing) {
        //si le champ de donnée n'est pas renseigné
        prenomErreur.textContent = "Veuillez indiquer votre prénom";
        prenomErreur.style.color = "red";
    }
}
prenom.onkeydown = function () {
    if (prenomRegexp.test(prenom.value) == true) {
        // si le champ est renseigné ET valide
        prenomErreur.textContent = "Format valide";
        prenomErreur.style.color = "lime";
    } else {
        // si le champ de donnée est incorrect
        prenomErreur.textContent =
            "Le format est incorrect (pas de chiffre ou caractères spéciaux)";
        prenomErreur.style.color = "orange";
    }
};
```

- **13 _ Gestion de l'envoi de la commande**

```
validation.addEventListener("click", (e) => {
  e.preventDefault();

  // recupération d'un tableau des id produit du localstorage
  let idArray = [];
  for (let i = 0; i < produitDuPanier.length; i++) {
    idArray.push(produitDuPanier[i].id);
  }

  //Rassembler les données à envoyer à l'API
  let commandeFinal = {
    // récupération des valeurs du formulaire dans l'objet contact
    contact: {
      firstName: prenom.value,
      lastName: nom.value,
      address: adresse.value,
      city: ville.value,
      email: email.value,
    },
    // et du tableau d'id
    products: idArray,
  };
```

```
if (
  prenomRegexp.test(prenom.value) == true &&
  nomRegexp.test(nom.value) == true &&
  emailRegexp.test(email.value) == true &&
  villeRegexp.test(ville.value) == true &&
  adresseRegexp.test(adresse.value) == true
) {
  //On envoie la commande au LS
  localStorage.setItem("commandefinal", JSON.stringify(commandeFinal));
  // on utilise post fetch pour envoyer la commande au serveur (API)
  fetch("http://localhost:3000/api/products/order", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Accept: "application/json",
    },
    body: JSON.stringify(commandeFinal),
  })
    .then((response) => response.json())
    .then((data) => {
      localStorage.clear();
      document.location.href = "confirmation.html?orderId=" + data.orderId;
    })
    .catch((err) => console.log(err.message));
} else {
  //...SINON
  return alert(
    "Merci de vérifier l'exactitude des renseignements du formulaire"
  );
}
```

# La page Confirmation - Objectifs

- L'utilisateur doit être redirigé vers la page de confirmation, une fois que la commande aura été validé avec succès

- Elle doit également lui indiquer son numéro de commande, qui correspond à l'orderId généré par l'API

# La page Confirmation - CODE

- Nous utilisons de nouveau URLSearchParams pour récupérer le numéro orderId qui se trouve dans l'URL, puis nous l'affichons dynamiquement

```
let orderId = new URLSearchParams(document.location.search).get("orderId");
document.getElementById("orderId").textContent = orderId;
```

# Le plan de test d'acceptation

+

# Démonstration