**READ ME (PRODUCTS)**

***Structure du Projet***

* *index.html* : Le fichier HTML principal contenant la structure de la page web.
* *prd.css*: Ce fichier contient les styles CSS pour la mise en page et le design de la page.
* *query.js:* Ce fichier contient les scripts JavaScript pour les interactions utilisateur et la manipulation du DOM.

***Structure HTML***

* *Header*: Inclut un logo et éventuellement d'autres éléments de navigation*.*
* *Section "Products*": Affiche une liste de produits avec des boutons de filtre.
* *Section "Shopping Cart":* Affiche le contenu du panier d'achat avec un bouton pour finaliser l'achat.
* *Footer:* Contient des liens vers les politiques de confidentialité, les produits disponibles et des informations sur l'entreprise*.*

***CSS et Style***

Le fichier CSS **(prd.css)** est lié dans la section **<head>** et gère le style de la page web, y compris la mise en page, les couleurs, la typographie et la réactivité.

***Fonctionnalité JavaScript***

Le fichier JavaScript (query.js) inclut les fonctionnalités suivantes :

1. *Affichage des produits :* Les produits sont récupérés à partir d'un fichier JSON et affichés sur la page principale du site.

const addDataToHTML = (filteredProducts) => {

$listProductHTML.empty();

if (filteredProducts.length > 0) {

filteredProducts.forEach(product => {

let newProduct = $(`

<div class="item" data-id="${product.id}">

<img src="${product.image}" alt="">

<h2>${product.name}</h2>

<div class="price">$${product.price}</div>

<button class="addCart">Add To Cart</button>

</div>

`);

$listProductHTML.append(newProduct);});}};

1. *Filtrage par catégorie :* Les produits peuvent être filtrés par catégorie à l'aide de boutons. Lorsqu'un bouton est cliqué, seuls les produits de la catégorie correspondante sont affichés.

const filterProductsByCategory = (category) => {

let filteredProducts;

if (category === 'all') {

filteredProducts = products;

} else {

filteredProducts = products.filter(product => {

return product.cat === category || (product.type && product.type.includes(category));});}

addDataToHTML(filteredProducts);};

$('.filter button').on('click', (event) => {

const category = $(event.target).attr('id');

filterProductsByCategory(category);});

1. *Ajout au panier* : Chaque produit affiché dispose d'un bouton "Add To Cart". Lorsque ce bouton est cliqué, le produit est ajouté au panier.

$listProductHTML.on('click', '.addCart', function() {

let id\_product = $(this).parent().data('id');

addToCart(id\_product);});

const addToCart = (product\_id) => {

let positionThisProductInCart = cart.findIndex((value) => value.product\_id == product\_id);

if (cart.length <= 0) {

cart = [{

product\_id: product\_id,

quantity: 1

}];

} else if (positionThisProductInCart < 0) {

cart.push({

product\_id: product\_id,

quantity: 1

});

} else {

cart[positionThisProductInCart].quantity += 1;

}

addCartToHTML();

addCartToMemory();

};

1. *Affichage du panier* : Le panier est affiché dans un élément HTML dédié. Il affiche les produits ajoutés, leur quantité et le prix total.

const addCartToHTML = () => {

$listCartHTML.empty();

let totalQuantity = 0;

let totalPrice = 0;

if (cart.length > 0) {

cart.forEach(item => {

totalQuantity += item.quantity;

let positionProduct = products.findIndex((value) => value.id == item.product\_id);

let info = products[positionProduct];

totalPrice += info.price \* item.quantity;

let newItem = $(`

<div class="item" data-id="${item.product\_id}">

<div class="image">

<img src="${info.image}">

</div>

<div class="name">

${info.name}

</div>

<div class="totalPrice">€${info.price \* item.quantity}</div>

<div class="quantity">

<span class="minus"><</span>

<span>${item.quantity}</span>

<span class="plus">></span>

</div>

</div>

`);

$listCartHTML.append(newItem);

});

}

$iconCartSpan.text(totalQuantity);

$('.total').text(`Total: €${totalPrice.toFixed(2)}`);

};

1. *Modification de la quantité dans le panier* : Pour chaque produit dans le panier, il est possible d'augmenter ou de diminuer la quantité à l'aide des boutons "<" et ">". Si la quantité atteint zéro, le produit est retiré du panier.

$listCartHTML.on('click', '.minus, .plus', function() {

let product\_id = $(this).closest('.item').data('id');

let type = $(this).hasClass('plus') ? 'plus' : 'minus';

changeQuantityCart(product\_id, type);

});

const changeQuantityCart = (product\_id, type) => {

let positionItemInCart = cart.findIndex((value) => value.product\_id == product\_id);

if (positionItemInCart >= 0) {

let info = cart[positionItemInCart];

switch (type) {

case 'plus':

cart[positionItemInCart].quantity += 1;

break;

default:

let changeQuantity = cart[positionItemInCart].quantity - 1;

if (changeQuantity > 0) {

cart[positionItemInCart].quantity = changeQuantity;

} else {

cart.splice(positionItemInCart, 1);

}

break;

}

}

addCartToHTML();

addCartToMemory();

};

1. *Stockage local du panier* : Les données du panier sont stockées localement dans le navigateur à l'aide de l'API localStorage. Cela permet de conserver le contenu du panier même après la fermeture du navigateur.

const addCartToMemory = () => {

localStorage.setItem('cart', JSON.stringify(cart));

};

1. *Interaction avec l'interface utilisateur* : L'interface utilisateur est interactive, avec des fonctionnalités telles que l'ouverture et la fermeture du panier en cliquant sur une icône de panier et un bouton de fermeture.

$iconCart.on('click', () => {

$body.toggleClass('showCart');

});

$closeCart.on('click', () => {

$body.toggleClass('showCart');

});

1. *Initialisation de l'application* : L'application est initialisée en récupérant d'abord les données des produits à partir d'un fichier JSON, puis en affichant les produits et le contenu du panier précédemment sauvegardé dans le localStorage.

const initApp = () => {

// get data product

$.getJSON('products.json', (data) => {

products = data;

addDataToHTML(products);

// get data cart from memory

if (localStorage.getItem('cart')) {

cart = JSON.parse(localStorage.getItem('cart'));

addCartToHTML();

}

});

};

initApp()

***Utilisation***

* Ouvrez le fichier index.html dans un navigateur web.
* Explorez les produits et utilisez les fonctionnalités de navigation et de panier.

***Remarque***

* Assurez-vous d'avoir une connexion internet active pour charger jQuery depuis un CDN.
* Ce projet peut nécessiter une connexion à un serveur pour des fonctionnalités avancées telles que la finalisation de l'achat.
* Vous pouvez acceder a cette page depuis la page d’accueil en cliquant sur products