Javascript cour

Les variables et constantes sont utilisées pour stocker des données. Dans le code, let et const sont utilisés pour déclarer des variables et des constantes.

* let cart = []; : Déclare une variable cart comme un tableau vide, qui stockera les articles ajoutés au panier.
* const cartSection = document.getElementById('cart-section'); : Utilise const pour déclarer une constante qui stocke une référence à un élément du DOM (Document Object Model).

**2. Sélection d'éléments du DOM**

Le code utilise plusieurs méthodes pour sélectionner des éléments du DOM.

* document.getElementById('view-cart-btn') : Sélectionne un élément en fonction de son ID.
* document.querySelectorAll('.add-to-cart') : Sélectionne tous les éléments ayant une classe spécifique.

**3. Écouteurs d'événements**

Les écouteurs d'événements permettent de réagir aux actions de l'utilisateur, comme les clics sur des boutons.

* button.addEventListener('click', function(event) {...}) : Ajoute un écouteur d'événement de clic à chaque bouton "Ajouter au panier".

**4. Manipulation du DOM**

Le code modifie le contenu des éléments du DOM en ajoutant ou en supprimant des éléments.

* cartItemsElement.innerHTML = ''; : Vide le contenu HTML d'un élément.
* const listItem = document.createElement('li'); : Crée un nouvel élément de liste (li).
* listItem.appendChild(removeButton); : Ajoute un élément enfant (removeButton) à un autre élément (listItem).

**5. Fonctions**

Les fonctions sont des blocs de code réutilisables. Elles permettent d'organiser et de structurer le code.

* function addToCart(productName, productPrice) {...} : Déclare une fonction qui ajoute un produit au panier.
* function updateCartDisplay() {...} : Déclare une fonction qui met à jour l'affichage du panier.

**6. Tableaux et Objets**

Les tableaux et objets sont des structures de données essentielles en JavaScript.

* cart.push({ name: productName, price: productPrice }); : Ajoute un objet représentant un produit au tableau cart.

**7. Boucles**

Les boucles permettent d'itérer sur des collections de données.

* cart.forEach((item, index) => {...}) : Parcourt chaque élément du tableau cart.

**8. Écouteur d'événements au chargement du DOM**

L'événement DOMContentLoaded garantit que le script s'exécute uniquement après le chargement complet du DOM.

* document.addEventListener('DOMContentLoaded', function() {...}) : Ajoute un écouteur d'événement pour exécuter le code après le chargement du DOM.

**9. Classes et Attributs**

Les classes et attributs HTML sont manipulés pour ajouter des styles ou des comportements spécifiques.

* navbar.classList.toggle('active'); : Bascule la classe active sur l'élément navbar.
* removeButton.setAttribute('data-index', index); : Définit un attribut data-index sur le bouton.

**10. Fonctions d'assistance**

Des fonctions d'assistance simplifient des tâches courantes, comme afficher un message de succès.

* function showSuccessMessage() { alert('Produit ajouté au panier avec succès !'); } : Affiche un message d'alerte lorsque le produit est ajouté au panier.

**Exemple d'exécution**

Lorsque l'utilisateur clique sur un bouton "Ajouter au panier", le produit correspondant est ajouté au tableau cart, un message de succès est affiché, et l'affichage du panier est mis à jour. Les éléments du panier peuvent être supprimés individuellement en cliquant sur les boutons "Supprimer", et le panier entier peut être vidé en cliquant sur le bouton "Vider le panier". Le bouton "Voir le panier" affiche la section du panier, et le bouton "Fermer le panier" la cache.

Ce code démontre comment JavaScript peut être utilisé pour créer une expérience utilisateur interactive et dynamique en manipulant le DOM, en réagissant aux événements de l'utilisateur et en utilisant des structures de données comme les tableaux et les objets.

/\*\*\*\*\*\*\*\*\*\*JS LE Dom\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Bien sûr ! Le Document Object Model (DOM) est une interface de programmation qui permet aux scripts (comme JavaScript) de mettre à jour le contenu, la structure et le style d'un document HTML ou XML. En termes simples, le DOM est une représentation en mémoire de la page web, qui permet aux développeurs de manipuler le contenu et l'apparence de la page de manière dynamique.

**Concepts de Base du DOM**

**1. Structure en Arborescence**

Le DOM représente le document sous la forme d'une arborescence de nœuds. Chaque élément HTML, attribut, et morceau de texte est un nœud dans cette arborescence.

* **Document** : Le nœud racine de l'arborescence. Par exemple, document en JavaScript représente l'ensemble du document HTML.
* **Éléments** : Les balises HTML sont des nœuds d'élément. Par exemple, <div>, <p>, <h1>, etc.
* **Attributs** : Les attributs des balises HTML sont des nœuds d'attribut. Par exemple, id="example" ou class="example".
* **Texte** : Le contenu texte à l'intérieur des éléments HTML est un nœud de texte.

**2. Accéder aux Éléments du DOM**

Pour manipuler les éléments du DOM, vous devez d'abord y accéder. Voici quelques méthodes courantes pour accéder aux éléments du DOM en JavaScript :

* document.getElementById(id) : Sélectionne un élément par son ID.

javascript

Copier le code

const header = document.getElementById('header');

* document.querySelector(selector) : Sélectionne le premier élément qui correspond à un sélecteur CSS.

javascript

Copier le code

const firstParagraph = document.querySelector('p');

* document.querySelectorAll(selector) : Sélectionne tous les éléments qui correspondent à un sélecteur CSS.

javascript

Copier le code

const allListItems = document.querySelectorAll('li');

**3. Manipuler les Éléments du DOM**

Une fois que vous avez accédé aux éléments du DOM, vous pouvez les manipuler en utilisant différentes méthodes et propriétés.

* **Modifier le contenu** :

javascript

Copier le code

const paragraph = document.getElementById('my-paragraph');

paragraph.textContent = 'Nouveau texte';

* **Modifier les attributs** :

javascript

Copier le code

const image = document.getElementById('my-image');

image.setAttribute('src', 'nouvelle-image.jpg');

* **Ajouter ou supprimer des classes** :

javascript

Copier le code

const element = document.querySelector('.my-element');

element.classList.add('nouvelle-classe');

element.classList.remove('ancienne-classe');

* **Créer et ajouter des éléments** :

javascript

Copier le code

const newElement = document.createElement('div');

newElement.textContent = 'Je suis un nouveau div';

document.body.appendChild(newElement);

* **Supprimer des éléments** :

javascript

Copier le code

const elementToRemove = document.getElementById('element-a-supprimer');

elementToRemove.remove();

**4. Écouteurs d'Événements**

Les événements permettent de rendre la page interactive en réagissant aux actions de l'utilisateur, comme les clics, les survols de souris, etc.

* **Ajouter un écouteur d'événement** :

javascript

Copier le code

const button = document.getElementById('my-button');

button.addEventListener('click', function() {

alert('Bouton cliqué !');

});

* **Supprimer un écouteur d'événement** :

javascript

Copier le code

function handleClick() {

alert('Bouton cliqué !');

}

button.removeEventListener('click', handleClick);

**Exemple Pratique**

Voici un exemple simple pour illustrer comment utiliser le DOM pour ajouter un nouvel élément à la page lorsqu'un bouton est cliqué :

HTML :

html

Copier le code

<!DOCTYPE html>

<html>

<head>

<title>Exemple DOM</title>

</head>

<body>

<button id="add-button">Ajouter un paragraphe</button>

<div id="content"></div>

<script src="script.js"></script>

</body>

</html>

JavaScript (script.js) :

javascript

Copier le code

// Sélectionner le bouton et la div content

const addButton = document.getElementById('add-button');

const contentDiv = document.getElementById('content');

// Ajouter un écouteur d'événement au bouton

addButton.addEventListener('click', function() {

// Créer un nouveau paragraphe

const newParagraph = document.createElement('p');

newParagraph.textContent = 'Nouveau paragraphe ajouté !';

// Ajouter le paragraphe à la div content

contentDiv.appendChild(newParagraph);

});

Dans cet exemple, lorsque le bouton "Ajouter un paragraphe" est cliqué, un nouveau paragraphe est créé et ajouté à la div content.

**Conclusion**

Le DOM est une interface puissante qui permet aux développeurs de créer des applications web interactives et dynamiques. En comprenant comment accéder aux éléments du DOM, les manipuler et réagir aux événements, vous pouvez créer des expériences utilisateur riches et engageantes.