**XMLHttpRequest**

XMLHttpRequest est un objet JavaScript qui permet de faire des requêtes HTTP (ou HTTPS) vers un serveur web. Il est principalement utilisé pour récupérer des données de manière asynchrone sans avoir à recharger la page. Voici un aperçu de ses caractéristiques et de son utilisation :

**Caractéristiques**

1. **Asynchrone** :
   * XMLHttpRequest permet d'envoyer des requêtes sans bloquer l'interface utilisateur. Cela signifie que l'utilisateur peut continuer à interagir avec la page pendant que la requête est en cours.
2. **Support des différents types de données** :
   * Il peut gérer différents types de données, comme JSON, XML, HTML, texte brut, etc.
3. **Compatibilité** :
   * Bien que XMLHttpRequest soit largement supporté dans tous les navigateurs modernes, il a été partiellement remplacé par l'API Fetch, qui offre une syntaxe plus simple et des fonctionnalités améliorées.

**Utilisation de base**

Voici un exemple simple d'utilisation de XMLHttpRequest :

javascript

Copier

// Créer une nouvelle instance de XMLHttpRequest

var xhr = new XMLHttpRequest();

// Configurer la requête

xhr.open('GET', 'https://api.example.com/data', true);

// Définir ce qui se passe lorsque la réponse est reçue

xhr.onload = function() {

if (xhr.status >= 200 && xhr.status < 300) {

// La requête a réussi, traiter la réponse

console.log(xhr.responseText);

} else {

// Gérer les erreurs

console.error('Erreur:', xhr.statusText);

}

};

// Gérer les erreurs de réseau

xhr.onerror = function() {

console.error('Erreur de réseau');

};

// Envoyer la requête

xhr.send();

**Étapes de l'exemple**

1. **Création de l'objet** :
   * var xhr = new XMLHttpRequest(); : On crée une nouvelle instance de XMLHttpRequest.
2. **Configuration de la requête** :
   * xhr.open('GET', 'https://api.example.com/data', true); : On configure la requête pour qu'elle soit de type GET et qu'elle cible une URL spécifique. Le troisième argument true indique que la requête est asynchrone.
3. **Gestion de la réponse** :
   * xhr.onload : Cette fonction est appelée lorsque la réponse est reçue. On vérifie le statut de la réponse pour déterminer si la requête a réussi.
4. **Gestion des erreurs** :
   * xhr.onerror : Cette fonction est appelée en cas d'erreur de réseau.
5. **Envoi de la requête** :
   * xhr.send(); : On envoie la requête au serveur.

**Conclusion**

XMLHttpRequest est un outil puissant pour effectuer des communications asynchrones dans les applications web. Bien qu'il soit moins utilisé aujourd'hui avec l'avènement de l'API Fetch, il reste essentiel pour comprendre le fonctionnement des requêtes HTTP en JavaScript.

**Structure du Code**

1. **Sélection des Éléments DOM** :

javascript

Copier

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

const content = document.getElementById("content");

* + Ces lignes sélectionnent des éléments HTML dans le document en utilisant leurs identifiants (id). header fait référence à l'élément avec l'ID header, et content à l'élément avec l'ID content.

1. **Définition de la Fonction Blague** :

javascript

Copier

function Blague() {

fetch("https://api.blablagues.net/?rub=blagues")

.then((res) => res.json())

.then((info) => {

console.log(info.data);

const blague = info.data.content;

header.textContent = blague.text\_head;

content.textContent = blague !== "" ? blague.text\_hidden : blague.text;

});

}

* + **fetch** : Cette fonction est utilisée pour faire une requête HTTP à l'URL spécifiée. Ici, elle interroge une API pour obtenir des blagues.
  + **.then((res) => res.json())** : Une fois que la réponse est reçue, elle est convertie en JSON. Cela permet d'extraire les données au format JavaScript.
  + **console.log(info.data)** : Affiche les données de la réponse dans la console pour le débogage.
  + **const blague = info.data.content** : On accède à la propriété content de l'objet data, qui contient les informations de la blague.
  + **Mise à jour du DOM** :
    - header.textContent = blague.text\_head; : Met à jour le contenu de l'élément header avec le titre de la blague.
    - content.textContent = blague !== "" ? blague.text\_hidden : blague.text; :
      * Cette ligne utilise un opérateur ternaire pour choisir quel texte afficher dans l'élément content. Si blague n'est pas vide, on affiche blague.text\_hidden, sinon on affiche blague.text.

1. **Appel Initial de la Fonction Blague** :

javascript

Copier

Blague();

* + Cela appelle la fonction Blague immédiatement lors du chargement de la page, ce qui permet d'afficher une blague dès le début.

1. **Événement sur le Bouton** :

javascript

Copier

btn.addEventListener('click', Blague);

* + Cette ligne ajoute un écouteur d'événements au bouton (btn). Lorsque le bouton est cliqué, la fonction Blague est appelée à nouveau, permettant de récupérer et d'afficher une nouvelle blague.

**Résumé**

* Le code effectue une requête à une API pour récupérer des blagues et les affiche dynamiquement dans la page web.
* Il utilise des promesses pour gérer les appels asynchrones et met à jour le DOM en fonction des données reçues.
* La fonction Blague est appelée à la fois lors du chargement de la page et lorsque l'utilisateur clique sur un bouton, permettant d'afficher une nouvelle blague à chaque clic.