5

# **Créer des requêtes Ajax en utilisant l’objet XMLHttpRequest en JavaScript**

Utiliser les gestionnaires d’événements load, error et progress

Nous allons pour cela utiliser des gestionnaires d’événements définis par l’interface XMLHttpRequestEventTarget qui vont nous permettre de prendre en charge différents événements déclenchés par notre requête et en particulier :

* L’événement load qui se déclenche lorsque la requête a bien été effectuée et que le résultat est prêt ;
* l’événement error qui se déclenche lorsque la requête n’a pas pu aboutir ;
* L’événement progress qui se déclenche à intervalles réguliers et nous permet de savoir où en est notre requête.

Au sein du gestionnaire d’événement load, on va déjà vouloir tester la valeur du statut code HTTP pour savoir si notre requête a bien abouti ou pas. Pour cela, nous allons observer la valeur de la propriété status de l’objet XMLHttpRequest.

Les statuts code HTTP les plus fréquents sont les suivants :

* 100 Continue : tout fonctionne jusqu’à présent; le client devrait continuer avec la requête ;
* 200 OK : Les requête a été un succès ;
* 301 Moved Permanently : L’identifiant de ressource unique (URI) relatif à la ressource demandée a changé de localisation de façon permanente ;
* 302 Found : L’identifiant de ressource unique (URI) relatif à la ressource demandée a changé de localisation de façon temporaire ;
* 304 Not Modified : Indique au client que la réponse n’a pas été modifiée depuis le dernier accès et qu’il peut utilisée la version en cache ;
* 401 Unauthorized : Indique que le client doit s’identifier s’il veut accéder à la réponse ;
* 403 Forbidden : Indique que le client n’a pas l’autorisation d’accéder à ce contenu ;
* 404 Not Found : Le serveur n’a pas pu trouver la ressource demandée ;
* 500 Internal Server Error : Le serveur a rencontré une situation qu’il ne peut pas gérer.

Ici, on va généralement tester si le statut code de notre réponse est bien égal à 200 en testant donc si la propriété status contient bien cette valeur. Si c’est le cas, on va pouvoir manipuler les données envoyées par le serveur.

Pour accéder à ces données, on va pouvoir utiliser la propriété response de l’objet XMLHttpRequest qui contient la réponse du serveur sous le format précisé par reponseType lors de l’envoi de la requête.

|  |
| --- |
| /\*Décommenter pour voir le résultat  //On crée un objet XMLHttpRequest  let xhr = new XMLHttpRequest();  //On initialise notre requête avec open()  xhr.open("GET", "une/url");  //On veut une réponse au format JSON  xhr.responseType = "json";  //On envoie la requête  xhr.send();  //Dès que la réponse est reçue...  xhr.onload = function(){  //Si le statut HTTP n'est pas 200...  if (xhr.status != 200){  //...On affiche le statut et le message correspondant  alert("Erreur " + xhr.status + " : " + xhr.statusText);  //Si le statut HTTP est 200, on affiche le nombre d'octets téléchargés et la réponse  }else{  alert(xhr.response.length + " octets téléchargés\n" + JSON.stringify(xhr.response));  }  };  //Si la requête n'a pas pu aboutir...  xhr.onerror = function(){  alert("La requête a échoué");  };  //Pendant le téléchargement...  xhr.onprogress = function(event){  //lengthComputable = booléen; true si la requête |

Une fois notre requête initialisée ou configurée grâce à open(), on va spécifier le format dans lequel le serveur doit nous renvoyer sa réponse en passant ce format en valeur de la propriété responseType de notre objet XMLHttpRequest. Les valeurs possibles sont les suivantes :

* "" (chaine de caractères vide) : valeur par défaut; demande au serveur de renvoyer sa réponse sous forme de chaine de caractères ;
* "text" : demande au serveur de renvoyer sa réponse sous forme de chaine de caractères ;
* "arraybuffer" : demande au serveur de renvoyer sa réponse sous forme d’objet ArrayBuffer ;
* "blob" : demande au serveur de renvoyer sa réponse sous forme d’objet Blob ;
* "document" : demande au serveur de renvoyer sa réponse sous forme de document XML ;
* "json" : demande au serveur de renvoyer sa réponse sous forme JSON.

Dans la majorité des cas, on demandera au serveur de nous renvoyer des données sous forme JSON (elles seront alors interprétées automatiquement) ou texte.

Une fois qu’on a défini le format de la réponse, nous allons pouvoir envoyer notre requête. Pour cela, nous allons utiliser la méthode send() de XMLHttpRequest.