Introduction à l'AJAX

Aurélien BOSSARD

1 Introduction

AJAX signifie Asynchronous Javascript And XML. C'est un ensemble de techniques utilisées pour communiquer entre le client et le serveur web, de manière asynchrone. Le client n'est donc pas bloqué en attendant la réponse du serveur aux requêtes envoyées en AJAX. Le X de XML est là pour nous rappeler qu'au début, XML était utilisé pour communiquer entre le client et le serveur. Aujourd'hui, le format de données JSON est de plus en plus utilisé à cet effet du fait de sa légèreté.

2 Installation de JQuery

Vous pouvez vous servir de la version de JQuery fournie par un CDN. Pour cela, il faut l'importer :

```
1 <script
2   src="https://code.jquery.com/jquery-3.6.1.min.js"
3   integrity="sha256-o88AwQnZB+VDvE9tvIXrMQaPlFFSUTR+nldQm1LuPXQ="
4   crossorigin="anonymous"></script>
```

L'avantage d'utiliser un CDN est la mise en cache côté client. Il est probable que le client ait déjà téléchargé cette version de JQuery en navigant sur un autre site. Son cache sera alors utilisé plutôt que de télécharger le fichier.

3 Premiers pas

JQuery est une bibliothèque qui permet de très facilement déployer des event listeners sur des éléments de l'arbre DOM. Ces éléments sont facilement sélectionnables via la syntaxe suivante, qui crée un objet en fonction d'un sélecteur css :

```
1 $('selecteur css')
```

Par exemple, je souhaite sélectionner toutes les balises a :

```
1 $('a')
```

Je souhaite sélectionner un élément dont l'id est test :

```
1 $('#test')
```

Je souhaite sélectionner tous les éléments dont la classe est test :

```
1 $('.test')
```

On peut également sélectionner des éléments grâce aux expressions XPath. Par exemple, je souhaite sélectionner tous les éléments avec un attribut test :

```
$('[test]')
```

Ou encore tous les éléments dont l'attribut test est égal à 1 :

```
$('[test="1"]')
```

On peut également utiliser des sélecteurs CSS beaucoup plus puissants, comme :first-child etc.

Nous allons réaliser un simple bouton qui va remplir un div lorsque l'on clique dessus. Pour cela, il faut trois choses :

1. Identifier le div sur lequel le bouton aura une action ;
2. Mettre en place un event listener sur le bouton ;
3. Écrire la fonction qui sera appelée par le trigger.

Pour identifier le div, on peut tout simplement lui donner un id, et allouer à notre bouton un attribut ad-hoc qui aura comme valeur l'id du div. Par exemple :

```
<div id="test">
</div>
<button targetID="test" id="boutonMagique">Clique-moi !</button>
```

Il faut ensuite définir un trigger sur le bouton. Les triggers déclenchent une fonction anonyme. Pour l'instant, pour une meilleure lisibilité, nous allons lui faire uniquement écrire dans les logs de la console :

```
$("#boutonMagique").click( //Ceci est la fonction anonyme
  function() {
    console.log("J'aime bien ca !");
  }
);
```

Placez ce script dans le body, après le bouton. Vérifiez que tout fonctionne bien en activant votre console de développement (F12 sous chrome) et en cliquant sur le bouton.

Maintenant, placez ce script dans un fichier js à part, et chargez-le juste après le chargement de jquery. Vous remarquerez que la fonction anonyme ne se déclenche plus lors d'un click sur le bouton. C'est parce que le script s'exécute et tente d'activer un event listener sur #boutonMagique alors que l'arbre DOM n'est pas complètement chargé. Pour exécuter le script après le chargement de l'arbre DOM, il faut utiliser une fonction anonyme qui se déclenchera après le chargement de l'arbre DOM :

```
$(
  function() {
    $("#boutonMagique").click( //Ceci est la fonction anonyme
      function() {
        console.log("J'aime bien ca !");
      }
    );
  }
);
```

On peut maintenant écrire dans le div. Pour cela, il suffit de le sélectionner avec un sélecteur jQuery et de changer son contenu (méthode html). Sur le manuel de jQuery, vous trouverez tous les attributs et méthodes applicables aux éléments. On peut tout d'abord construire un String qui contiendra le sélecteur css de notre div. Pour cela, il faut récupérer l'attribut "targetID" de notre bouton. Dans la fonction anonyme d'un event listener, l'élément sur lequel l'événement a été déclenché s'appelle avec $(this) . Un

attribut d'un élément se récupère avec la méthode attr, qui prend en paramètre le nom de l'attribut dont on veut récupérer la valeur. On peut alors construire notre sélecteur css pour le div d'id "boutonMagique" et changer son contenu en appelant dessus la méthode html, qui prend en paramètre le contenu à placer dedans.

```
1  $(
2    function() {
3      $("#boutonMagique").click( //Ceci est la fonction anonyme
4        function() {
5          var selecString = "#"+$(this).attr("targetID");
6          $(selecString).html("J'aime bien ca !");
7        }
8      );
9    }
10 );
```

Attention ! Si jamais vous voulez ajouter des éléments à la volée, qui eux-mêmes disposent d'un event listener : par exemple, un formulaire sur lequel on ajoute à la volée des inputs et des boutons liés à ces inputs. Dans ce cas, si vous aviez défini un event listener avec un sélecteur css sur la même classe que celle des boutons ajoutés à la volée, cet event listener ne s'appliquera pas aux nouveaux éléments ajoutés. Il faudra redéfinir un event listener sur les nouveaux éléments "à la main".

4 Communication avec le serveur web

Maintenant que nous avons défini notre event listener sur notre bouton, nous voulons que lorsque l'événement est déclenché, le client envoie une requête asynchrone au serveur afin de recevoir une réponse qu'il placera dans le div.

Pour cela, nous utiliserons la méthode ajax de jQuery. Cette méthode appellera une page distante. Dans le cadre de votre projet web, il faut garder en tête que de faire de l'ajax nous place dans une architecture différente de celle de votre projet. Par conséquent, il faudra, pour chaque page qui sera appelée pour répondre à un besoin spécifique en ajax, créer une nouvelle page indépendante de l'index. Vous pouvez en revanche réutiliser les classes que vous aurez déjà définies et qui peuvent vous aider.

Voici comment appeler une page distante à l'aide de la méthode ajax. Vous trouverez plus de détails dans le manuel de jQuery. data sert à définir les données qui seront envoyées à la page distante, ici avec la méthode POST. dataType correspond au type de retour de la page distante.

```
1  $.ajax({
2    method: "POST",
3    url: "test.php",
4    data: { param1 : "blabla", param2: "bliblibli" },
5    dataType : "json"
6  })
7    .done(function( retour ) {
8      alert( "Reponse : " + retour );
9          //On peut maintenant utiliser retour comme un objet jQuery
10   });
```

La page distante (test.php) devra spécifier qu'elle retourne du json, grâce à la ligne suivante :

```
1 header('Content-Type: application/json; charset=utf-8');
```

Puis elle devra effectuer les traitements et renvoyer les objets/tableaux nécessaires au Javascript côté client jsonifiés à l'aide de la fonction json_encode. La méthode ajax côté client convertira automatiquement le json en objets / tableaux utilisables en javascript.

5 Exercice

Utilisez les différentes techniques vues précédemment pour réaliser :

- Un input dans lequel un utilisateur peut taper un texte
- Un bouton qui :
  - Récupère les données de l'input
  - Les transmet à une page côté serveur
  - Récupère la réponse de la page côté serveur et la place dans un div déjà créé et vide au départ

La page côté serveur devra tester si le texte envoyé est présent dans une colonne d'une table de la base de données, et renvoyer un message en fonction.

Vous pouvez faire varier les event listener pour, par exemple, appeler le script lorsqu'un caractère est entré dans l'input, lorsque le focus de l'input est perdu, etc...