AJAX JQuery

Qu'est-ce qu'AJAX ?

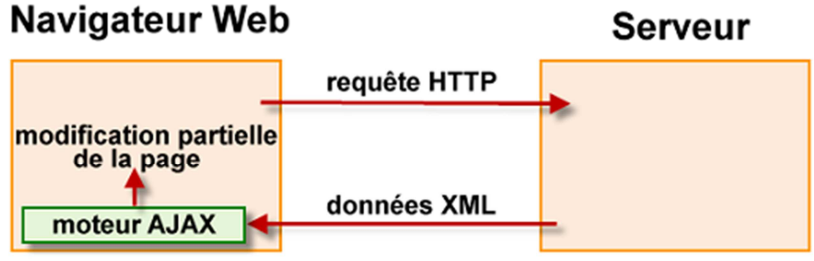
Lorsque vous naviguez de page en page sur un site Web traditionnel (entendez par là non-AJAX), les actions de l'internaute se traduisent par les actions suivantes :

1. Envoi d'une requête au serveur afin d'obtenir une nouvelle page.
2. Calcul de la nouvelle page par le serveur et envoi des données HTML/CSS correspondantes.
3. Affichage de ces données dans le navigateur.

Cette technique fonctionne très bien dans la plupart des cas, mais parfois seule une partie de la page nécessite d'être mise à jour. C'est là qu'intervient AJAX :

1. Dans un premier temps, envoi d'une requête au serveur afin d'obtenir les données qui seront affichées dans une partie bien précise de la page actuelle.
2. Calcul des données demandées par le serveur et envoi de ces données au navigateur au format XML.
3. Réception des données envoyées par le programme (on dit aussi moteur) AJAX qui les a demandées et affichage dans un endroit bien précis de la page actuelle sans toucher au reste de la page.

La figure suivante résume ces deux modes de fonctionnement.



Si, dans la plupart des cas, un fonctionnement traditionnel est entièrement satisfaisant, les performances d'affichage peuvent être grandement améliorées dans certains cas particuliers, comme par exemple l'affichage de données mises à jour à intervalles réguliers (cours d'actions en bourse par exemple), la sauvegarde des données pendant la saisie dans un formulaire, la mise à jour et/ou la vérification dynamique des champs d'un formulaire en fonction des données saisies par l'utilisateur, la saisie prédictive (comme le fait Google en proposant des réponses lorsque vous commencez à taper quelques caractères dans la case de recherche), etc.

Que signifie le terme AJAX ? AJAX est l'acronyme d'*Asynchronous JavaScript and XML*. Tous ces termes se comprennent aisément : le langage JavaScript est utilisé pour demander des données au serveur. Ces données lui sont retournées de façon asynchrone sous une forme XML.

Les méthodes se différencient des fonctions, car elles s'appliquent à des éléments obtenus à travers un sélecteur jQuery. Dans $('sélecteur').meth(paramètres);, meth est une méthode, alors que dans $.fonc(paramètres);, fonc est une fonction.

Charger des données JSON

Charger des données codées en JSON

JSON (*JavaScript Object Notation*) est un format de données textuel qui permet de représenter des informations structurées. Voici un exemple de données au format JSON :

```
{
  'menu': 'Fichier',
  'commande': [
      {
          'nomCde': 'Nouveau',
          'action':'CreateDoc'
      },
      {
          'nomCde': 'Ouvrir',
          'action': 'OpenDoc'
      },
      {
          'nomCde': 'Enregistrer sous',
          'action': 'SaveAs'
      }
      {
          'nomCde': 'Fermer',
          'action': 'CloseDoc'
      }
   ]
}
```

Comme vous pouvez le déduire en examinant ce code, un fichier JSON est composé d'un ensemble de paires 'nom':'valeur' organisées de façon hiérarchique. Ici par exemple, les noms menu et commande se trouvent au même niveau hiérarchique. Quant aux noms nomCde et action, il s'agit des enfants du nom commande.

Vous serez peut-être amenés à manipuler des données au format JSON. Pour cela, vous chargerez le fichier de données JSON avec la fonction $.getJSON(), puis vous travaillerez sur les différentes données qui le composent en utilisant la fonction de rappel.

Pour bien comprendre comment accéder aux données d'un fichier codé en JSON, nous allons raisonner sur un exemple simple qui comporte quatre paires 'nom':'valeur' de même niveau :

```
{
  "nom": "Pierre Durand",
  "age": "27",
  "ville": "Paris",
  "domaine": "HTML5, CSS3, JavaScript"
}
```

Voici le code HTML/jQuery utilisé pour manipuler ces données :

```
<button id="charger">Charger et traiter les données</button>
<div id="r">Cliquez sur "Charger et traiter les données" pour lancer la lecture et le
traitement des données JSON</div>

<script src="jquery.js"></script>
<script>
  $(function() {
    $('#charger').click(function() {
        $.getJSON('fichier.json', function(donnees) {
        $('#r').html('<p><b>Nom</b> : ' + donnees.nom + '</p>');
        $('#r').append('<p><b>Age</b> : ' + donnees.age + '</p>');
        $('#r').append('<p><b>Ville</b> : ' + donnees.ville + '</p>');
        $('#r').append('<p><b>Domaine de compétences</b> : ' + donnees.domaine + '</p>');
      });
    });
  });
</script>
```

Lorsque le bouton est cliqué, la fonction getJSON() est exécutée pour charger le fichier de données fichier.json :

```
$.getJSON('fichier.json', function(donnees) {
```

Le deuxième paramètre de la fonction getJSON() correspond à la fonction de rappel. Cette fonction est exécutée lorsque le fichier de données a été entièrement chargé. Remarquez le mot donnees passé comme paramètre de la fonction de rappel. C'est par son intermédiaire que les données JSON seront accessibles.

Dans un premier temps, la valeur correspondant au nom (donnees.nom) est extraite du fichier de données et placée sous une forme HTML (html()) dans la balise #r. Comme nous passons par la méthode html() pour remplir la balise , il est possible d'utiliser des attributs de mise en forme. Ici, le mot « Nom » est mis en gras avec la balise HTML <b> :

```
$('#r').html('<p><b>Nom</b> : ' + donnees.nom + '</p>');
```

La donnée age (donnees.age) est alors extraite du fichier de données et placée à la suite du nom, dans un nouveau paragraphe. Ici aussi, le nom du champ est mis en gras en utilisant la balise HTML <b>.

Deux instructions similaires extraient les données ville et domaine du fichier de données JSON et les affichent à la suite du nom et de l'âge :

```
$('#r').append('<p><b>Ville</b> : ' + donnees.ville + '</p>');
$('#r').append('<p><b>Domaine de compétences</b> : ' + donnees.domaine + '</p>');
```

La fonction $.ajax()

La fonction $.ajax() permet d'envoyer des requêtes HTTP AJAX à un serveur Web avec beaucoup de finesse dans les paramètres qui peuvent lui être communiqués.

```
$.ajax(adresse, {options});

$.ajax({options});
```

adresse est l'adresse à laquelle la requête doit être envoyée, et options correspond à une ou plusieurs des options suivantes :

- type : type de la requête, GET ou POST (GET par défaut).
- url : adresse à laquelle la requête doit être envoyée.
- data : données à envoyer au serveur.
- dataType : type des données qui doivent être retournées par le serveur : xml, html, script, json, text.
- success : fonction à appeler si la requête aboutit.
- error : fonction à appeler si la requête n'aboutit pas.
- timeout : délai maximum (en millisecondes) pour que la requête soit exécutée. Si ce délai est dépassé, la fonction spécifiée dans le paramètre error sera exécutée.

Beaucoup d'autres options peuvent être utilisées. Pour en avoir une liste exhaustive, consultez la documentation officielle.

Pour comprendre le fonctionnement de cette fonction nous allons utiliser l'exemple suivant :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Ajax - La fonction ajax()</title>
  </head>
```

```
  <body>
    <button id="action">Lancer la requête AJAX</button><br />

    <script src="jquery.js"></script>
    <script>
      $(function() {
        $('#action').click(function() {
          $.ajax({
            type: 'GET',
            url: 'proverbes.php?l=7',
            timeout: 3000,
            success: function(data) {
              alert(data); },
            error: function() {
              alert('La requête n\'a pas abouti'); }
          });
        });
      });
      </script>
  </body>
</html>
```

Le corps du document contient un bouton de commande qui sera utilisé pour exécuter la requête AJAX. Lorsque ce bouton est cliqué, la fonction $.ajax() est lancée pour exécuter la requête AJAX. Cette requête est de type GET (ce paramètre aurait pu être omis, puisqu'il s'agit de la valeur par défaut). La page invoquée est définie dans le paramètre url et le délai maximal d'exécution l'est dans le paramètre timeout. Ici, un délai de 3000 millisecondes est accordé au programme PHP pour fournir ce qui lui est demandé.

Si la requête aboutit, les données renvoyées par le programme PHP sont affichées dans une boîte de message. Dans le cas contraire, un message d'erreur est affiché.