

TP Ajax JQuery

Version : 1.0

Le TP sera consacré à la création d'un système d'auto-complétion qui sera capable d'aller chercher, en AJAX par le biais de la fonction `$ajax()`, les villes de France commençant par les lettres que vous aurez commencé à écrire dans le champ de recherche.

Présentation de l'exercice

Les technologies à employer

Avant de commencer, il nous faut déterminer le type de technologie dont nous avons besoin, car ici nous ne faisons pas uniquement appel au Javascript, nous allons devoir employer d'autres langages.

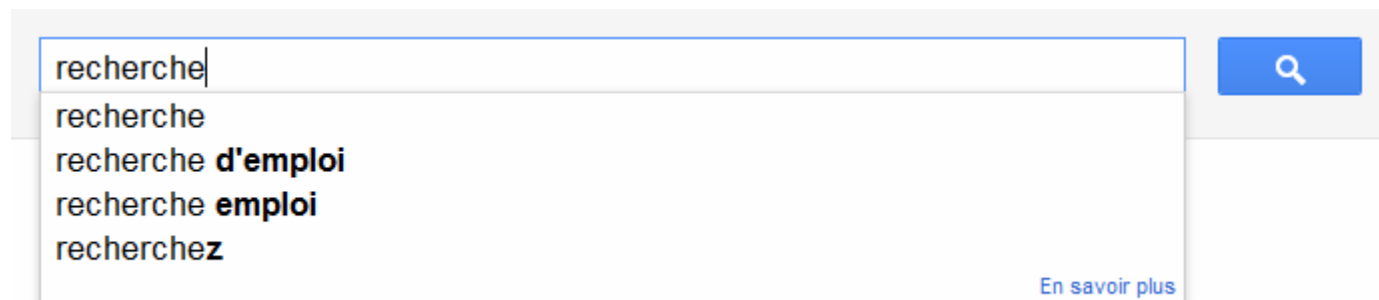
Dans un cadre général, un système d'auto-complétion fait appel à trois technologies différentes :

- Un langage client ayant la capacité de dialoguer avec un serveur ;
- Un langage serveur capable de fournir les données au client ;
- Une base de données qui stocke toutes les données.

Dans notre cas, nous allons utiliser le JavaScript et JQuery pour le langage client et le PHP pour le langage serveur. Nous allons, en revanche un fichier de stockage plutôt qu'une base de données afin de simplifier notre code.

Principe de l'auto-complétion

Un système d'auto-complétion se présente de la manière suivante :



The image shows a search interface. At the top, there is a light gray bar containing a search input field and a blue search button with a magnifying glass icon. The input field contains the text "recherche". Below the input field, a white dropdown menu is open, displaying a list of suggestions: "recherche", "recherche d'emploi", "recherche emploi", and "recherchez". The text "recherche d'emploi" and "recherche emploi" are highlighted in blue. In the bottom right corner of the dropdown menu, there is a link that says "En savoir plus" in blue text.

Google a mis en place un système d'auto-complétion sur son moteur de recherche

Conception

L'interface

L'auto-complétion étant affiliée, généralement, à tout ce qui est du domaine de la recherche, il va falloir un champ de texte pour écrire les mots-clés. Cependant, ce dernier va nous poser problème, car le navigateur enregistre généralement ce qui a été écrit dans les champs de texte afin de vous le reproposer plus tard sous forme d'auto-complétion, ce qui va donc faire doublon avec notre système. Il est possible de désactiver cette auto-complétion en utilisant l'attribut `autocomplete` de cette manière :

```
<input type="text" autocomplete="off" >
```

À cela nous allons devoir ajouter un élément capable d'englober les suggestions de recherches. Celui-ci sera composé d'une balise `<div>` contenant autant de `<div>` que de résultats, comme ceci :

```
<div id="results">
  <div>Résultat 1</div>
  <div>Résultat 2</div>
</div>
```

Chaque résultat dans les suggestions devra changer d'aspect lorsque celui-ci sera survolé ou sélectionné par le biais des touches fléchées.

À titre d'information, puisque vous allez devoir gérer les touches fléchées afin de naviguer dans les résultats, voici les valeurs respectives de la propriété `keyCode` pour les touches `Haut` , `Bas` et `Entrée` : 38, 40 et 13.

La communication client/serveur

Nous devons effectuer une requête à chaque caractère écrit afin de proposer une liste de suggestions. Il nous faudra donc une fonction liée à l'événement `keyup` de notre champ de recherche, qui sera chargée d'effectuer une nouvelle requête à chaque caractère tapé.

Cependant, si nous tapons deux caractères dans le champ de recherche, que la première requête réponde en 100 ms et la seconde en 65 ms : nous allons alors obtenir les résultats de la première requête *après* ceux de la seconde et donc afficher des suggestions qui ne seront plus du tout en accord avec les caractères tapés dans le champ de recherche. La solution à cela est simple : utiliser la méthode `abort()` sur la précédente requête effectuée si celle-ci n'est pas terminée. Ainsi, elle ne risque pas de renvoyer des informations dépassées par la suite.

Le traitement et le renvoi des données

Côté serveur, nous allons faire un script de recherche basique dont le principe consiste à rechercher les villes qui correspondent aux lettres entrées dans le champ de recherche. Nous allons charger un

fichier dans lequel se trouve un tableau PHP linéarisé contenant les plus grandes villes de France, il ne restera plus qu'à l'analyser.

Le fichier towns.txt est à récupérer sur e-campus.

La linéarisation d'une variable en PHP permet de sauvegarder des données sous forme de chaîne de caractères. Cela est pratique lorsque l'on souhaite sauvegarder un tableau dans un fichier, c'est ce qui a été fait pour les villes. Les fonctions permettant de faire cela se nomment [serialize\(\)](#) et [unserialize\(\)](#).

Il faut récupérer les données contenues dans le fichier towns.txt. Pour cela, il faut lire le fichier avec la fonction [file_get_contents\(\)](#), puis convertir son contenu en tant que tableau PHP grâce à la fonction [unserialize\(\)](#).

Une fois cela fait, le tableau obtenu doit être parcouru à la recherche de résultats en cohérence avec les caractères tapés par l'utilisateur dans le champ de recherche. Pour cela, vous aurez besoin d'une boucle ainsi que de la fonction [count\(\)](#) pour obtenir le nombre d'éléments contenus dans le tableau.

Pour récupérer les mots-clés de la recherche nous utiliserons la méthode GET et un nom de champ « s », ce qui nous donne la variable PHP `$_GET['s']`.

Pour vérifier si un index du tableau correspond à votre recherche, il va vous falloir utiliser la fonction [stripos\(\)](#), qui permet de vérifier si une chaîne de caractères contient certains caractères, et ce sans tenir compte de la casse. Si vous trouvez un résultat en cohérence avec la recherche, alors ajoutez-le à un tableau (que vous aurez préalablement créé) grâce à la fonction [array_push\(\)](#).

Une fois le tableau parcouru, il ne reste plus qu'à trier les résultats avec la fonction [sort\(\)](#), puis à renvoyer les données au client.

Les données seront renvoyées au format JSON qui est un format de données textuel de plus en plus utilisé.

Il existe une fonction PHP qui vous permet de convertir un tableau au format JSON : il s'agit de la fonction [json_encode\(\)](#). Une fois la fonction utilisée, il ne reste plus qu'à retourner le tout au client avec echo et à analyser cela côté JavaScript.

Commencer par coder le script serveur, car il est possible d'analyser manuellement les données retournées par le serveur, et ce sans avoir déjà codé le script client. On peut donc s'assurer du bon fonctionnement du serveur avant de s'attaquer au client.

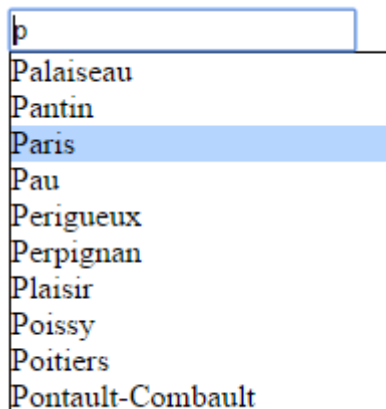
Une fois le code du serveur écrit et testé, écrivez le code HTML, qui se sera extrêmement simple avec un champ de texte sur lequel nous avons désactivé l'auto-complétion ainsi qu'une balise <div> destinée à accueillir la liste des résultats obtenus :

```
<input id="search" type="text" autocomplete="off" />
<div id="results"></div>
```

Nous pouvons maintenant coder le JavaScript. L'événement utilisé sur le champ input search est keyup et va se charger de gérer les interactions entre l'utilisateur et la liste de suggestions. Il doit permettre, par exemple, de naviguer dans la liste de résultats et d'en choisir un avec la touche **Entrée**, mais il doit aussi détecter quand le contenu du champ de recherche change et alors faire appel au serveur pour obtenir une nouvelle liste de résultats.

Pour chaque déplacement de la sélection, il vous faut appliquer un style sur le résultat sélectionné afin que l'on puisse le distinguer des autres. Vous pouvez affecter une classe CSS lorsqu'un résultat est sélectionné et la retirée dès qu'un autre résultat est sélectionné.

Au final vous devez avoir un résultat :



A screenshot of a web interface showing a search input field. The input field contains the letter 'p'. Below the input field, a dropdown menu is open, displaying a list of suggestions. The suggestions are: Palaiseau, Pantin, Paris, Pau, Perigueux, Perpignan, Plaisir, Poissy, Poitiers, and Pontault-Combault. The 'Paris' suggestion is highlighted with a blue background.

Search Input
p

Suggestions
Palaiseau
Pantin
Paris
Pau
Perigueux
Perpignan
Plaisir
Poissy
Poitiers
Pontault-Combault