Le référencement de l'Ajax

Retour au sommaire de la lettre

Domaine :	Recherche	**Référencement**
Niveau :	Pour tous	**Avancé**

L'Ajax est une technologie très intéressante pour accélérer l'affichage de pages pour lesquelles seule une partie du contenu doit être mise à jour lors de l'affichage. Malheureusement, son incompatibilité avec Google et les autres moteurs de recherche la rend complexe à mettre en oeuvre et à référencer. Voici donc trois solutions techniques pour s'affranchir de ces obstacles et obtenir une meilleure indexation de vos contenus générés grâce à l'Ajax...

La vitesse et l'ergonomie sont des atouts dont un site ne peut plus se passer aujourd'hui, tant au niveau de l'expérience utilisateur que du référencement naturel. Pour optimiser ces éléments de manière simultanée, il est donc possible d'utiliser de l'Ajax : l'*Asynchronous Javascript and XML*.

Cette technologie permet de charger de manière dynamique des contenus, sans pour autant recharger l'intégralité de la page. Cela réduit donc les temps de chargement, tout en donnant une sensation de navigation plus fluide pour l'utilisateur.

Le problème en SEO, c'est que l'Ajax n'est pas compris par Google et les autres moteurs de recherche : autrement dit, tout contenu chargé avec cette technologie est invisible, à moins de prendre les bonnes mesures pour les faire indexer.

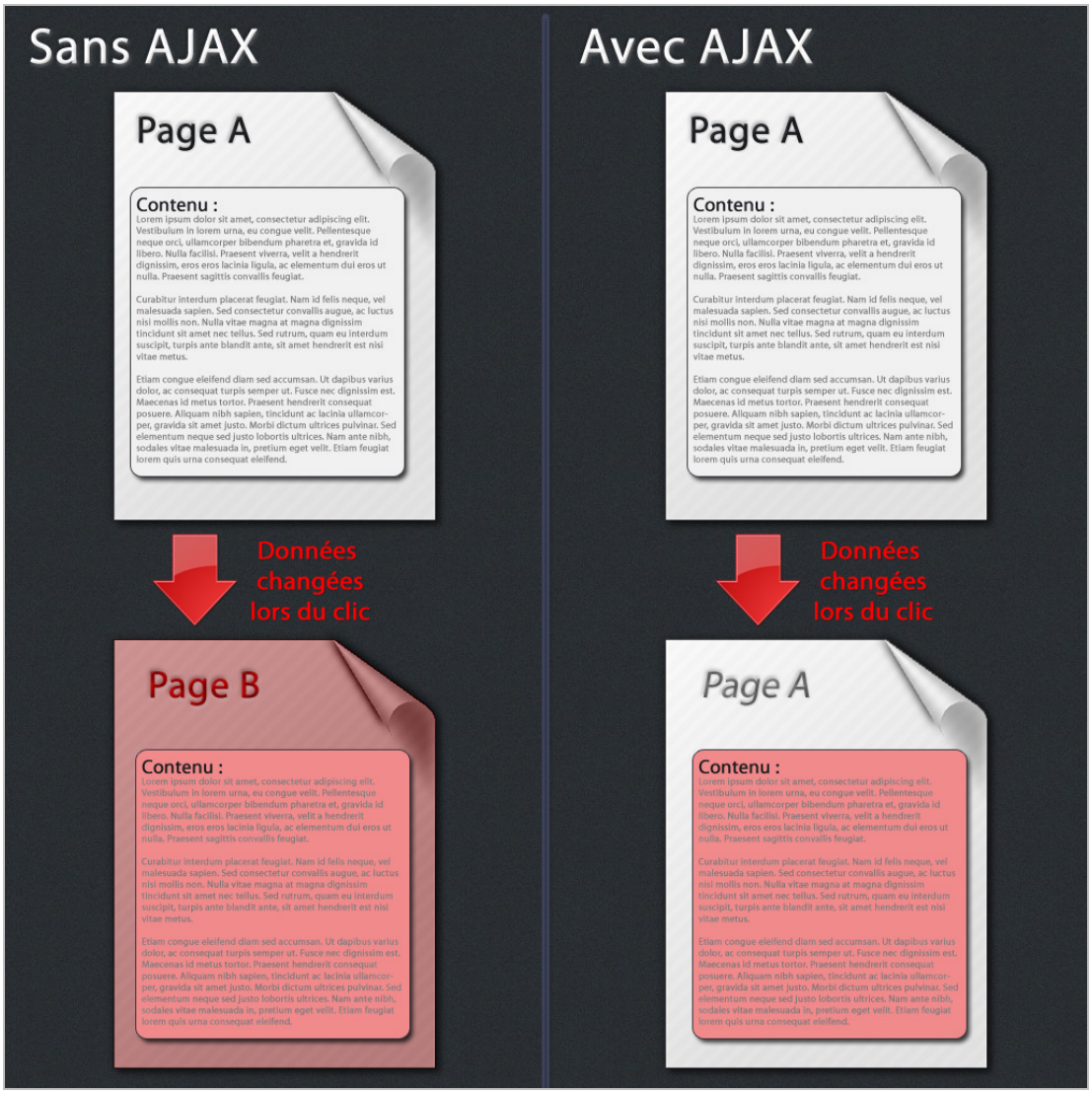
L'Ajax, c'est quoi ?

Pour mieux comprendre la raison pour laquelle l'Ajax pose autant de problèmes en référencement naturel, il faut tout d'abord expliquer le fonctionnement de base d'un site Internet lors du chargement des pages.

De base, l'ordinateur de l'internaute envoie une requête au serveur Internet, avec pour paramètre une URL et éventuellement des cookies. Le serveur calcule alors la page dans son intégralité et la renvoie à l'utilisateur. La page est donc entièrement recalculée, y compris au niveau des parties communes d'un site comme le haut de page, le footer et parfois les colonnes de droite ou de gauche. On va donc recalculer et télécharger des éléments que l'utilisateur avait déjà mis en cache dans son navigateur.

L'Ajax est une technologie "tampon". Avec elle, seule une partie de la page sera calculée par le serveur. Par exemple, l'utilisateur va envoyer une demande en Ajax pour charger le contenu d'une nouvelle page, sauf que les parties communes de celle-ci ne changeront pas : on ne va télécharger que la partie pertinente ce qui va réduire drastiquement les temps de chargement.

L'image suivante explique le fonctionnement de cette technologie, en mettant en rouge les éléments qui sont chargés par l'internaute lors du clic.



On observe donc facilement que les contenus Ajax sont de fait moins lourds, donc plus rapides à charger. D'un certain côté, on peut rapprocher le fonctionnement de l'Ajax des antiques "frames", utilisées il y a quelques années de cela pour construire des sites web...

Le problème de l'indexation de l'Ajax

De base, la technologie Ajax ne change que le contenu demandé par le script : théoriquement, il ne modifie ni l'URL, ni les balises métas. Google et les autres moteurs de recherche ne peuvent donc pas trouver ce contenu. Autrement dit, une page généré en Ajax est catastrophique pour le référencement naturel...

La difficulté est donc de mettre en place des liens en dur, du type http://www.monsite.com/mapage, qui vont charger le contenu en Ajax dans la page actuelle, tout en changeant l'URL et en dirigeant le moteur de recherche vers la page adéquate.

La dernière partie est primordiale, mais souvent oubliée ou mal conçue par les développeurs. On retrouve ainsi des sites superbes et fluides pour les internautes, mais qui sont aussi pertinents pour les moteurs de recherche qu'une page blanche... Heureusement, il existe trois solutions différentes pour pallier le problème :

1. La solution Javascript

La première solution pour référencer ce type de contenus est simple. On commence par créer son site de manière purement statique, avec des liens en durs pour aller d'un page à une autre page.

On va dans un second temps ajouter une surcouche de Javascript qui va s'appliquer à tous les liens interne, et dans laquelle on indique au navigateur de remplacer une partie du contenu actuel par celui de la page demandée. On réussit donc à charger en Ajax un contenu pour le visiteur, tandis que le moteur de recherche continuera à voir un site statique ayant un maillage et une structure interne cohérente. Pour cela, on utilisera un code javascript ressemblant à celui-ci :

```
$('#IDdulien').live('click',function(e){
        // On empêche le navigateur de charger le lien
e.preventDefault();
        //On récupère la valeur du lien
var centercible = $(this).attr('href');
        // On sélectionne les données à charger en Ajax
var loadUrl="data.html";
        // On indique où il faut afficher ces nouvelles données
                $("#ID").load(loadUrl + "ID");
        //On change l'URL selon les données contenues dans le Href du lien
                window.location=centercible;
 });
```

Les avantages de cette solution sont multiples :

- Elle est simple à mettre en place.
- Google continuera à suivre chaque lien normalement, sans pénalité pour le référencement naturel.
- C'est parfaitement transparent pour l'utilisateur.
- Même si le javascript est désactivé dans le navigateur, l'utilisateur pourra continuer à utiliser normalement le site Internet.

Malheureusement, cette technique a un gros défaut qui se situe dans l'URL du navigateur : un site en Ajax ne peut pas changer réellement l'adresse web sur laquelle se trouve l'utilisateur, du moins s'il est codé en HTML4 (nous verrons plus loin que l'HTML5 permet de corriger le problème).

Dans le meilleur des cas, le script va ajouter une ancre, pour obtenir une adresse du type www.monsite.com/#ancre-du-contenu-ajax. Cet ajout du signe # va servir uniquement à l'utilisateur, afin qu'il comprenne qu'il change de page lors du clic (sans réellement en changer pour autant). En effet, l'Ajax ne peut changer complètement une URL sans recharger la page dans son intégralité. Ainsi, il n'est pas possible de passer en Ajax de www.monsite.com/url1 à www.monsite.com/url2, mais seulement à www.monsite.com/url1#contenu-url2.

L'internaute se trouve donc en présence d'une URL générée côté client et non côté serveur, ce qui pose un problème majeur : le visiteur ne pourra pas partager l'adresse par email, sur Facebook, sur Twitter ou encore sur son propre site. Si jamais il fait un copier-coller de l'adresse web de la page, celle-ci ne contiendra pas le bon contenu puisque l'ancre et le contenu associé n'apparaissent que lors du clic, ce que ne sait faire un moteur de recherche.

On pourrait bien entendu corriger le problème (pour le visiteur uniquement), en détectant l'ancre au chargement de la page et en adaptant le contenu si besoin est, mais impossible de faire cela pour Google qui n'exécutera pas le code. Et même si c'était faisable pour un moteur de recherche, cela surchargerait énormément les pages...

Autrement dit, la solution de base est incomplète :

- Il est impossible de partager l'URL en Ajax.
- Les boutons sociaux présents sur la page doivent être codés pour constamment récupérer la bonne URL pour fonctionner.
- Il existe une possibilité de rendre possible le partage de l'URL par l'internaute, mais cela surcharger le javascript qui doit détecter l'ancre au chargement de la page

2. La solution du HeadLess Browser

Pour pallier l'ensemble de ces défauts, il existe une solution donnée par le moteur de recherche Google lui-même : la mise en place sur son serveur d'un *HeadLess Browser* (un navigateur sans interface).

Le principe est simple : le site Internet ajoute un signe supplémentaire à chacune des URLs en Ajax: un point d'exclamation. Notre adresse devient alors #! au lieu du simple #, comme c'est actuellement le cas sur Twitter. Cela indique à Google qu'il doit faire appel au *HeadLess browser* pour indexer le rendu déjà généré de la page. Autrement dit, le moteur de recherche va pouvoir visualiser le contenu Ajax comme le ferait un visiteur lors de sa navigation.

En réalité, Google transforme la partie "#!ID" avec ses propres paramètres, soit une URL du type ?_escaped_fragment_=ID. Cela lui indique d'utiliser la technologie du *HeadLess browser*. Le seul changement à réaliser pour le développeur web sur le *front-end* du site est donc de bien vérifier que chaque contenu en Ajax ajoute le point d'exclamation en plus du #. Il faudra ensuite configurer Apache pour également rediriger le visiteur vers le bon contenu lorsque celui-ci vient sur le site avec une adresse du type #!mon-contenu.

Les avantages sont évidents :

- Le site est toujours aussi simple à développer en front-end puisqu'il faut juste ajouter un point d'exclamation.
- Cela permet le partage et l'indexation des URLs en Ajax pour l'ensemble des visiteurs et des moteurs de recherche.

C'est au niveau du serveur que cela se complique, car il faut configurer correctement son serveur Apache, installer et configurer Jetty ainsi qu'une Webapp. Cela nous amène donc directement aux inconvénients du HeadLess browser :

- Il nécessite un hébergement compatible sur lequel on peut modifier facilement la configuration du serveur, ce qui exclut un grand nombre de sites en mutualisé.
- Il nécessite également une personne suffisamment compétente pour le mettre en place.
- La solution ne fonctionne que pour Google, et pas pour Yahoo et Bing.

La procédure est en effet relativement longue et complexe. Elle n'est pas du tout destinée aux néophytes, et il vous faudra avoir un minimum de connaissances en administration de serveur pour implanter cette solution. Elle se décompose en plusieurs étapes :

- L'ajout du point d'exclamation pour les contenus en Ajax.
- Installer Jetty, qui va permettre de déployer notre application Web.
- Installer grâce à Jetty le HeadLess Browser.
- Activer un proxy sur son serveur, qu'il faut ensuite protéger des spammeurs.
- Mettre en place une réécriture d'URL pour prendre en compte l'*escaped_fragment* pour les visiteurs et les moteurs de recherche.

Pour faire tout cela, nous vous invitons donc à suivre le tutoriel disponible à cette adresse :
http://www.seomix.fr/referencement/naturel/ajax-headless-browser/

Google a également créé une page officielle d'explication ici :
http://code.google.com/intl/fr/web/ajaxcrawling/docs/getting-started.html

3. La solution Html5

Puisque la première solution est incomplète et que la seconde pose problème quant à son implantation, il faut trouver une solution plus complète. Elle existe : c'est l'HTML5.

Depuis quelques temps, cette nouvelle mouture (encore en développement) de l'HTML introduit de nouvelles possibilités pour le développement web. D'ailleurs, de plus en plus de sites Internet l'adoptent et les navigateurs s'adaptent de mieux en mieux à cette technologie. Autrement dit, il y a de moins en moins de risques à développer directement son site Internet

en HTML5, d'autant plus qu'il existe des scripts de compatibilité pour les anciens navigateurs et que Google parvient parfaitement à indexer cette version de l'HTML.

Ce qui fait la différence par rapport à la version 4 au niveau de l'Ajax, c'est qu'il existe désormais une fonction pour modifier en dur l'URL du navigateur, sans recharger la page. Mieux encore, ce changement permet de créer un historique de navigation, ce qui permet à l'utilisateur d'utiliser à volonté les boutons "suivant" et "précédent".

Cette fonction "miracle" s'appelle **history.pushState**, et va nous permettre donc de modifier facilement l'URL sur laquelle se trouve l'utilisateur. Elle prend la forme suivante :

```
history.pushState(data, title, url);
```

En utilisant ce code, on indique au navigateur quelles données (*data*) sont associées à quel titre de page (*title*) pour une URL donnée (*url*). Cela va donc inscrire ces informations dans l'historique de navigation de l'utilisateur, lui permettant ainsi de naviguer librement.

Parfois, la donnée "Data" de *history.pushState* est vide (ce qui peut arriver si votre script est mal conçu ou si votre animation de page est trop spécifique). Même dans ce cas, il existe une solution de secours si l'utilisateur cherche à appuyer sur le bouton "page précédente". Cette solution consiste à vérifier si l'utilisateur fait appel à cette fonctionnalité, et d'exécuter alors une fonction que vous aurez codé vous-même :

```
window.addEventListener("popstate", function(e) {
        //ma-fonction-de-retour
}
```

Nous n'avons donc plus besoin d'Ancres du type # ou #!, et encore moins d'un *HeadLess browser*. Tout comme la première solution qui a été donnée, le site est codé en dur, et c'est une surcouche javascript qui va gérer tous les contenus en Ajax. Les liens en dur sont donc corrects pour les visiteurs comme pour les moteurs de recherche. Notre site est donc parfaitement indexable, et optimisé pour le référencement naturel.

Certains sites ont déjà mis en place cette technologie, comme le site https://github.com/ qui permet à ses utilisateurs de partager et d'échanger du code source, des plugins et des hacks. Quand on est dans un projet, toute la navigation en Ajax pour passer d'un fichier à un autre fait appel à la fonction *history.pushState*.

Pour ceux que cela intéresse, voici un excellent article qui explique comment implanter une telle solution technique : http://diveintohtml5.org/history.html

Cependant, comme n'importe quelle technologie récente, cette fonctionnalité de l'HTML 5 reste incompatible avec certains navigateurs, comme avec Internet Explorer (toutes versions confondues), Opéra Mini ainsi que Safari et Safari mini. Heureusement pour nous, des développeurs ont créé un javascript permettant de simuler la fonction *history.pushState* dont nous avons besoin. Il suffit de faire appel au script disponible à cette adresse : http://plugins.jquery.com/project/history-js

Conclusion

L'Ajax est une technologie vraiment intéressante pour les visiteurs, qui leur donne une réelle sensation de vitesse et d'ergonomie. Malheureusement, cette technologie pose problème pour l'indexation et le référencement naturel, malgré les différentes solutions qui existent. D'ailleurs, ces méthodes ne fonctionnent qu'avec Google : Bing, et donc Yahoo!, restent malheureusement à la traîne...

Il ne nous reste donc qu'à attendre encore quelques années, que les navigateurs utilisés par les internautes soient plus adaptés à l'HTML 5, ou bien que Google se décide enfin à exécuter les javascripts présents sur les pages qu'il indexe. C'est pourtant techniquement faisable, mais le moteur de recherche le plus ancré en France devrait investir massivement dans des centaines de serveurs supplémentaires pour exécuter puis indexer l'ensemble de ces contenus en Ajax... En a-t-il vraiment envie aujourd'hui ?

Daniel Roch, *Consultant WordPress, Référencement et Webmarketing chez SeoMix (http://www.seomix.fr/)*

Réagissez à cet article sur le blog des abonnés d'Abondance :
http://blog-abonnes.abondance.com/2011/09/le-referencement-de-lajax.html