## Contenu potentiellement visible : ce que Googlebot voit vraiment...

Retour au sommaire de la lettre

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

*Depuis 16 ans, le moteur de recherche Google a grandement amélioré sa perception du contenu d'une page web telle que la voit l'inetrnaute : interpétation du Javascript, de l'Ajax, détection du cloaking, prise en compte des différentes parties d'une page web, responsive design, ainsi que le contenu "potentiellement visible" (onglets, contenu en accordéon, etc.). L'objectif du robot du moteur est donc bien de "voir" une page web, telle que l'internaute l'a sous les yeux, et de l'anayser sous cette forme. Qu'en est-il aujourd'hui et jusqu'où va le moteur dans cette perspective ?...*

Le 24 novembre 2014, John Mueller, l'un des porte-parole de Google, lâchait lors de l'un ses fameux « hangouts », une confirmation qui a surpris de nombreux observateurs : placer des contenus importants dans des zones non visibles par défaut par un utilisateur (dans une zone caché activée par un onglet par exemple) peut s'avérer être une mauvaise idée si on veut être bien classé sur un mot clé figurant dans ce contenu.

Beaucoup ont cru que cette déclaration signifiait que Google n'indexait plus le contenu non visible par défaut par un utilisateur… En réalité il n'en est rien. Et Google est aujourd'hui capable de crawler et d'indexer bien plus de choses que vous ne le pensez peut être. Nous sommes entrés depuis plusieurs années déjà dans une phase où Googlebot connait très bien l'apparence de vos pages aux yeux de vos utilisateurs, et a commencé depuis quelque temps déjà à en tenir compte pour définir l'importance des contenus dans une page.

La « révélation » de John Mueller est donc l'occasion de rappeler à quel point les capacités du spider de Google, Googlebot, ont progressé depuis le lancement du moteur de recherche…

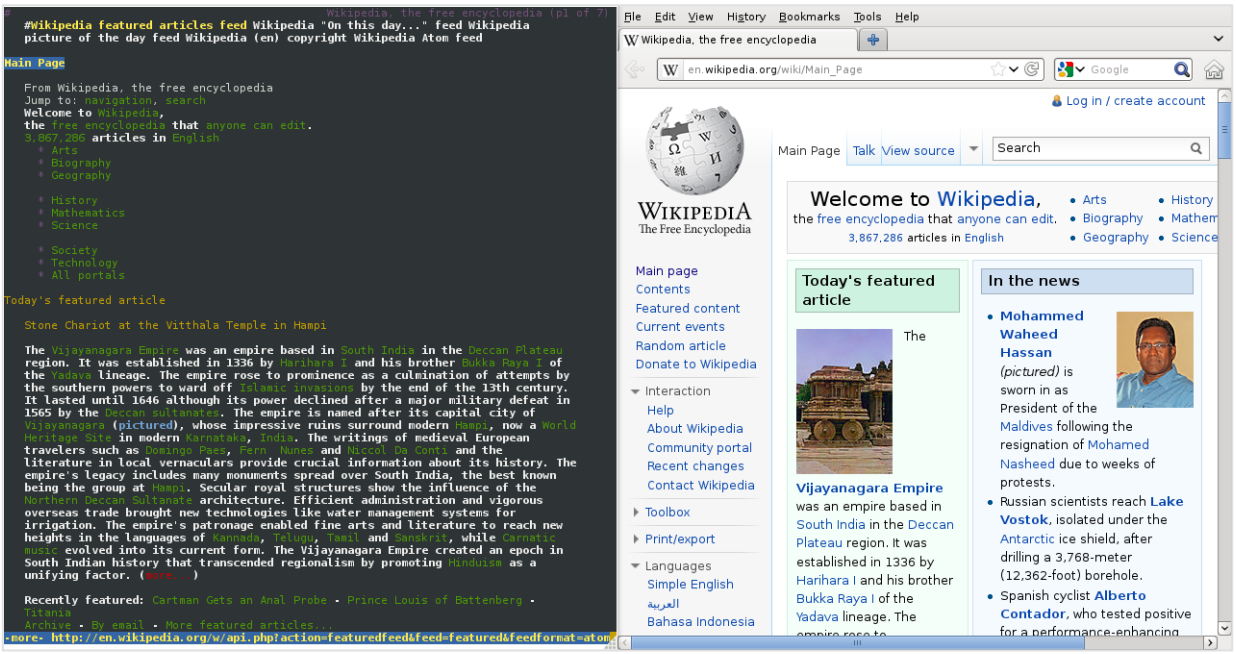
### *L'époque où Google avait un oeil de... Lynx*

Lors des débuts de Google en 1998, le crawler Googlebot se comportait de manière extrêmement basique : il téléchargeait le code HTML, sans se préoccuper de ressources externes comme les feuilles de style et les fichiers de code javascript. Les images étaient récupérées par un crawler spécialisé, et faisaient l'objet d'un processus séparé.

Le contenu des pages HTML était ensuite « parsé » (analysé) pour séparer le contenu textuel du code HTML définissant sa présentation dans un navigateur. Seules quelques balises HTML étaient prises en compte dans la phase d'indexation, soit pour constituer les snippets (comme la balise title, ou la meta description), soit pour donner un poids plus important aux termes entourés par des balises indiquant un contenu important ou mis en avant (comme les balises Hn, les balises <B> ou , par exemple).

Mais dans tous les cas, avant que le contenu soit entièrement décortiqué pour être indexé terme par terme, tout le code HTML était volontairement « oublié » par les processus du moteur de recherche…

Bref tout se passait comme si Google avait une vision extrêmement limitée des pages HTML… Pendant longtemps, les spécialistes utilisaient un navigateur open source ultra simplifié, baptisé « Lynx », pour simuler le contenu que Google pouvait réellement voir à l'aide de Googlebot. Et effectivement, un coup d'œil sur les résultats obtenus permettaient de se rendre compte des capacités « visuelles » extrêmement limitées de Googlebot.




*Comparaison d'une page de wikipedia « vue » à gauche à l'aide de l'outil Lynx, et à droite avec Firefox. On se rend compte à quel point la compréhension d'une page par un spider peut être limitée.*

## Parser le javascript, les flash et les autres documents pour y trouver des liens

Cette vision extrêmement rudimentaire du contenu d'une page web s'est révélé rapidement poser des problèmes sérieux au moteur. L'une des premières évolutions visibles fût de permettre au « spider » de Google de parser tous les formats de fichiers pouvant contenir des liens hypertextes pour y découvrir de nouvelles ressources web à explorer : notamment les liens placés dans des fichiers flash, dans du code javascript (à condition qu'ils soient « en clair » sous une forme reconnaissable de type http://www.domaine.com/page.html...) ou des documents word, powerpoint etc…

## Détecter le cloaking

Après quelques années de fonctionnement du moteur, les webmasters attentifs se sont rendu compte que des crawlers dotés d'IP de Google aspiraient leur page avec un user agent classique de type Mozilla. Bien sûr, dans certains cas, il s'agissait d'employés de Google surfant normalement sur leurs sites, mais le caractère systématique de certaines visites ont révélé qu'il pouvait aussi s'agir de tests pour voir si les sites utilisaient une pratique dite de « cloaking » : c'est-à-dire présenter un contenu différent à un navigateur classique et à Googlebot (une technique prohibée par les *guidelines* du moteur de recherche).

Mais notons que ce n'est pas parce que l'on utilise un user-agent de type « Mozilla Firefox » que l'on transforme son « spider » en un navigateur complet… Et dans les premières années de la décennie 2000, cette tactique pour détecter les fraudeurs continuait de s'appuyer sur un crawler ultra simplifié.

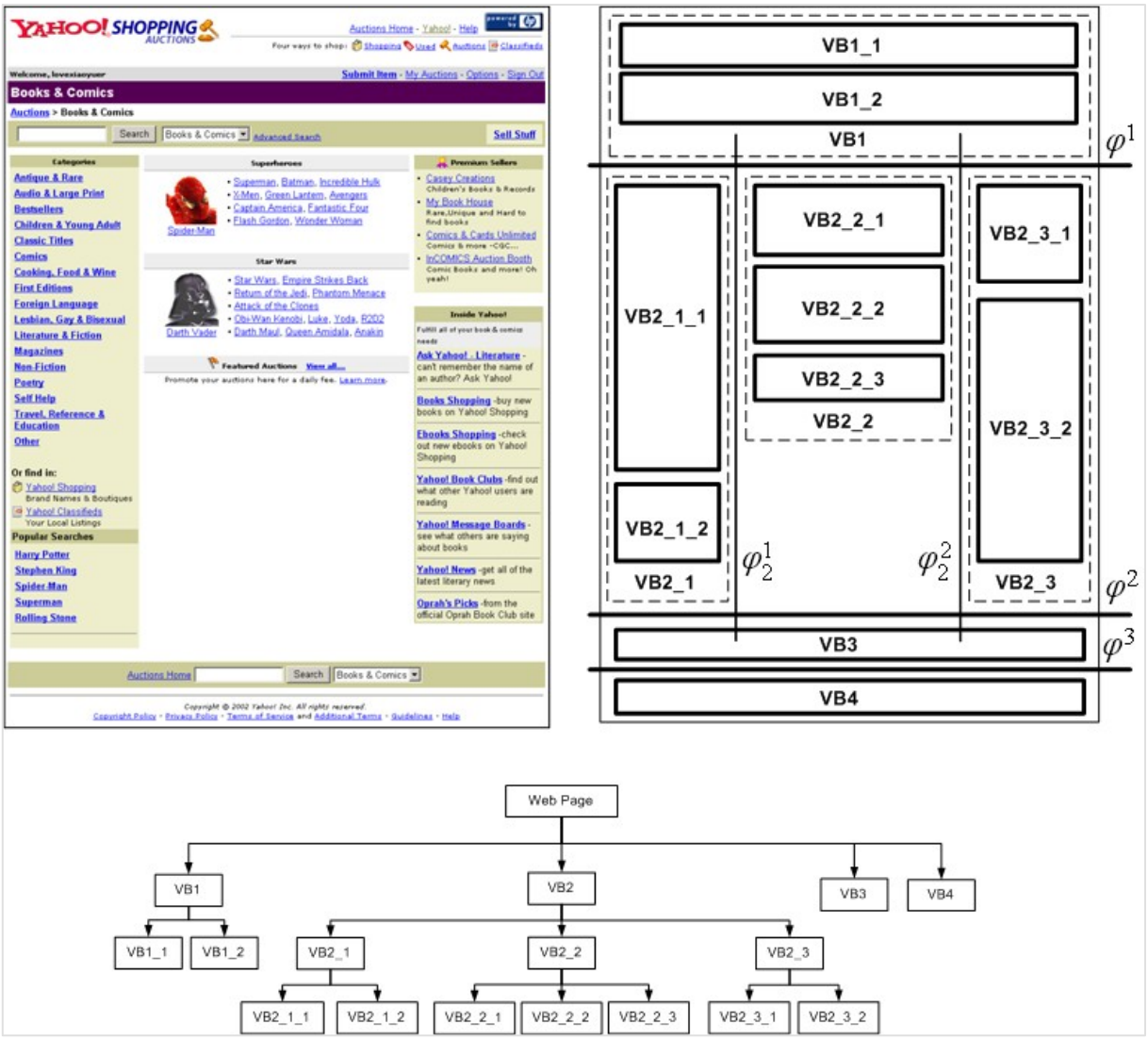
## Isoler le contenu unique des boilerplates et comprendre l'apparence générale de la page

Dès 2003, les ingénieurs travaillant dans les équipes de recherche des principaux moteurs de recherche grand public ont commencé à ressentir le besoin d'aller plus loin dans la compréhension de l'apparence d'une page web pour améliorer la pertinence de leurs algorithmes.

En particulier, les équipes de Microsoft ont publié plusieurs articles scientifiques présentant des méthodes pour tenir compte du zonage d'une page web afin d'identifier les contenus importants qui y figuraient.

Ils proposaient en particulier d'analyser le code HTML et les feuilles de style pour :
- Séparer le contenu spécifique à une page des « boilerplates », c'est-à-dire les zones communes à de nombreuses pages du site comme les header, footer, colonnes de menu à gauche, barres de menu etc.
- Et identifier l'agencement des zones sur la page.

En particulier, ils remarquaient dans leurs articles que les contenus placés en bas d'une colonne située dans la partie la plus en bas à droite de la page avaient une forte probabilité d'être des contenus publicitaires ou peu importants, tandis que la zone centrale était la plus susceptible de contenir des éléments importants.




*Une illustration issue de l'article des chercheurs de Microsoft, décrivant le système VIPS : la page crawlée est analysée pour identifier les différents blocs composant la page. Ces informations sont ensuite exploitées pour donner plus ou moins de poids aux liens figurant dans ces blocs, en fonction de leur emplacement sur la page.*

Cette approche, consistant à surpondérer ou sous-pondérer les éléments en fonction de leur emplacement réel sur une page, a été depuis lors reprise également par les ingénieurs de Google.

## *Exécuter le Javascript, appliquer les css : Googlebot devient un navigateur (presque) complet*

Mais une difficulté subsistait encore : l'apparence d'une page dans un navigateur n'est pas uniquement définie par le code HTML et les feuilles de style. La « rendition » finale dépend aussi du résultat du code Javascript contenu dans les pages et exécuté par le navigateur. L'apparition et l'adoption de navigateurs de plus en plus puissants et sophistiqués par les internautes, ainsi que l'évolution des normes HTML et CSS, a conduit en parallèle à une utilisation de plus en intensive du Javascript dans les pages web.

Par contre, exécuter le Javascript pour se faire une idée de l'apparence « réelle » d'une page web (celle que voit un internaute) est un défi technique.

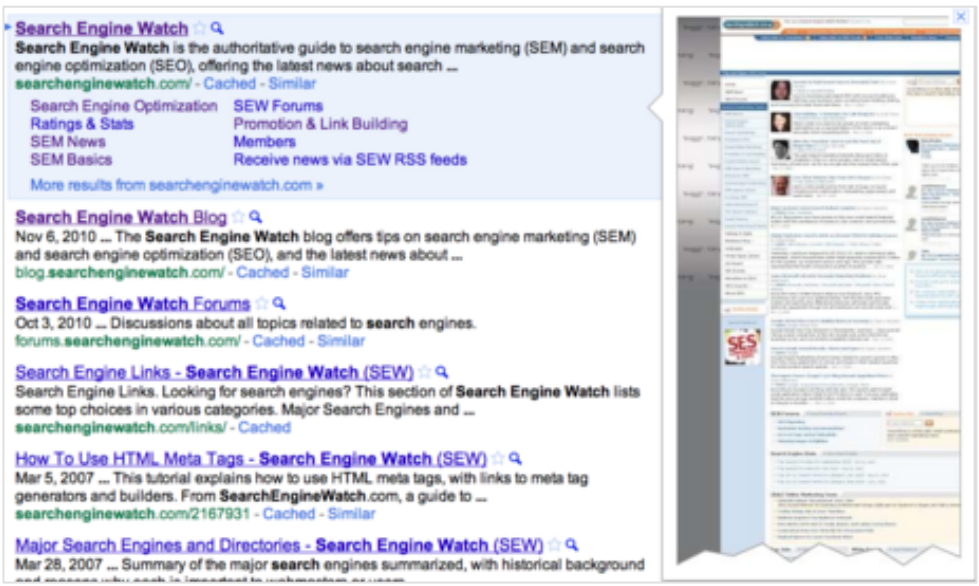
La principale difficulté ne réside pas dans l'ajout de cette fonctionnalité au processus de crawl : depuis une dizaine d'années, on trouve des moteurs Javascript simplifiés open source, capables d'exécuter le code à des fins d'exploration du web ou pour découvrir du contenu généré en Javascript, et qui peuvent parfaitement être « greffés » à un robot d'exploration dans un moteur de recherche.

Non, le problème numéro 1 reste qu'exécuter du code Javascript est très consommateur de ressources informatiques… La puissance de calcul nécessaire pour réussir à exécuter le code Javascript de toutes les pages crawlées par Google est simplement phénoménale. Pendant de nombreuses années, l'idée d'exécuter le Javascript a été écartée car trop consommatrice de ressources pour le gain de pertinence apporté, et même parce qu'elle était susceptible de « ralentir » considérablement le processus de crawl et d'indexation de Google.

Avec le temps et l'évolution des technologies, l'augmentation de la puissance des processeurs et de la mémoire vive disponibles sur les machines utilisées par Google a rendu le ratio coût/bénéficie de l'exécution des Javascripts beaucoup plus favorable.

On soupçonne Google d'avoir effectué ses premiers tests dès les années 2006 / 2007 (peut-être même avant ?).

En 2009, Google lance les « instant previews » : chaque page web présentée dans les pages de résultat se voit dotée d'une image montrant la « rendition » complète de la page. Cette fonctionnalité nécessite l'emploi d'un « moteur de rendition » complet, capable d'exécuter le Javascript, et ce… à l'échelle de toutes les pages indexées. Notons que cette fonctionnalité a pu être déployée indépendamment des processus de crawl et d'indexation du contenu textuel, et que l'on ne sait pas si Google a exploité dès cette époque les informations fournies par le moteur de rendition.




*Un aperçu des « instant previews » proposées par Google entre fin 2010 et avril 2013*

En 2011, Matt Cutts a confirmé que Google avait commencé à exécuter le code javascript des pages dans certaines circonstances, à des fins d'indexation. Et en mai 2014, un billet du blog pour les webmasters de Google a annoncé que cette fonctionnalité était officiellement systématiquement employée. A tel point que les guidelines pour les webmasters ont été modifiées en octobre dernier, afin de déconseiller formellement d'empêcher le crawl des ressources Javascript et css externes via le robots.txt (ce qui peut évidemment empêcher Google de parvenir à la même rendition que le browser d'un internaute).

## *Le problème de l'Ajax et du chargement asynchrone, et des contenus cachés derrière des requêtes en POST*

En novembre 2011, Matt Cutts révèle dans un tweet que Google a encore amélioré les capacités de son crawler, notamment pour parvenir à indexer certains contenus affichés en Ajax. Un billet dans le blog pour les webmasters de Google a précisé dans la foulée que les avancées concernaient non seulement la prise en compte de l'Ajax, mais aussi une capacité renforcée d'exploration de contenus accessibles via des requêtes en mode POST.


## *C'est officiel : Googlebot sait exécuter le javascript*

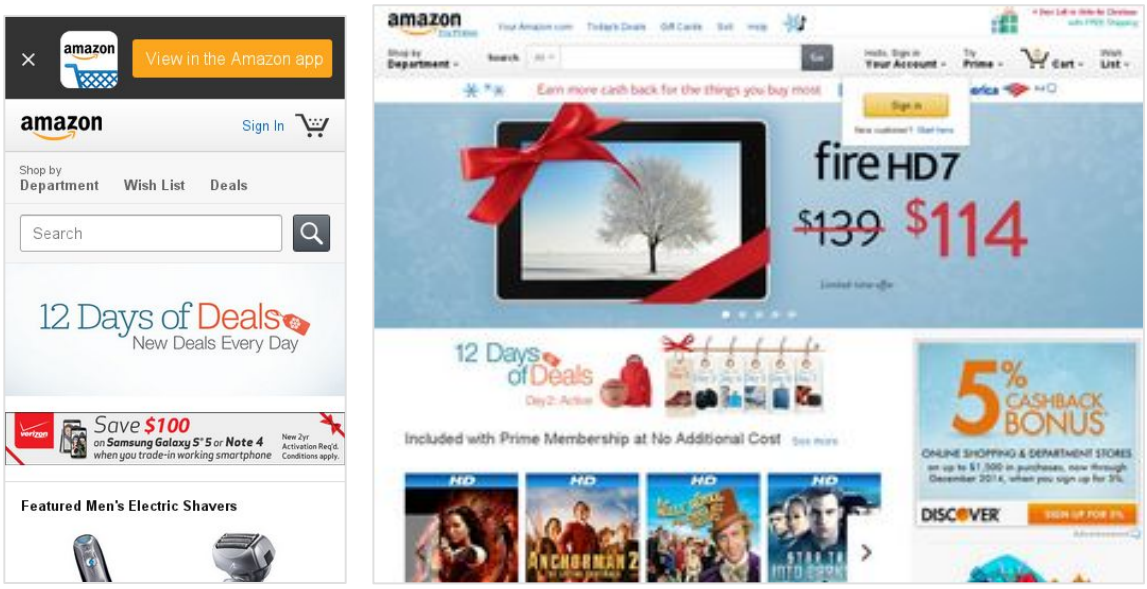
La dernière étape dans cette longue évolution des capacités de Googlebot a été bien sûr l'annonce, le 23 mai dernier, *via* un billet du blog pour les webmasters que Google savait exécuter le Javascript. Le billet continuait néanmoins de mettre en garde les propriétaires de site sur les problèmes que peuvent engendrer, pour la bonne prise en compte des contenus, la présence dans le code de code Javascript bogué ou trop complexe.

## *Googlebot, les devices et le responsive design*

Cette évolution des capacités du spider de Google était indispensable, car aujourd'hui une part importante des pages web fait un usage très important de code en Javascript.

C'est d'autant plus vrai avec la montée en puissance parallèle des usages sur smartphones, tablettes : le "responsive design" pose de nouveaux défis. A terme, il va devenir indispensable pour les moteurs de recherche de tenir compte des différentes renditions, pour comprendre :
- Ce qui est visible ou pas dans les différentes configurations ;
- En particulier les liens disponibles dans les différentes versions (mobiles, tablettes, desktop)…

On s'achemine à moyen terme vers une différenciation croissante entre les classements affichés sur les différentes versions des moteurs de recherche (mobile ou desktop) en fonction de critères mesurés ou identifiés sur les renditions spécifiques à chaque version.




*Previews de la page d'accueil d'Amazon générées par Google PageSpeed Insights (à gauche version mobile, à droite version desktop). Le contenu et la navigation dans les deux contextes est très différent : les moteurs de recherche ont besoin de tenir compte de ces différences pour continuer à retourner des résultats pertinents*

## Le contenu « caché » est-il toujours pris en compte ?

Depuis quelques semaines, des observateurs ont noté que leurs pages ne parvenaient plus à se positionner normalement sur des contenus affichés dans des zones masquées aux utilisateurs par défaut (comme des onglets supplémentaires, ou des zones extensibles - "accordéons" - en cliquant sur une icone).

Une petite phrase lâchée par John Mueller lors d'un hangout le 24 novembre dernier en réponse à une question sur ce point a intrigué les observateurs attentifs : *« From our point of view, it's always a tricky problem when we send a user to a page where we know this content is actually hidden. Because the user will see perhaps the content in the snippet, they'll click through the page, and say, well, I don't see where this information is on this page. I feel kind of almost misled to click on this to actually get in there.*
*So that's kind of the problem that we're seeing…. I think we've been picking up on that for quite some time now to kind of discount that information. It might be that we've gone a little bit further now to actively ignore the information that's not directly visible. »*

Traduction : *« de notre point de vue, c'est toujours problématique d'envoyer un utilisateur sur une page alors que nous savons que le contenu est en fait caché. Parce que l'utilisateur verra peut-être le contenu dans le snippet, il cliquera sur le résultat et se dira, bon, je ne vois pas où est cette information sur cette page. On se sent en quelque sorte presque abusé en cliquant là-dessus pour arriver là…*
*Donc c'est le type de problèmes que nous rencontrons… Je pense que nous avons relevé cela depuis un certain temps déjà, et que nous écartons ce type d'informations. Il est possible que nous ayons été un peu plus loin maintenant, en ignorant volontairement les informations qui ne sont pas visibles. »*

Que faut-il réellement en conclure ? Tout d'abord, et c'est assez facile à vérifier, que les textes des snippets, lorsqu'ils reprennent des contenus de la page web, sont effectivement choisis préférentiellement dans des zones visibles par défaut par l'utilisateur.

Ensuite, John Mueller nous indique qu'en sus de ce comportement, qui existe depuis un bon moment déjà, Google a commencé à donner un poids moindre, voire à ignorer totalement les contenus cachés.

Pour le moment, des tests simples montrent que ce contenu continue d'être crawlé et indexé, et que ce changement de comportement n'est pas du tout généralisé. Mais cela peut changer rapidement.

## Quelles conclusions en tirer pour le SEO ?

Le comportement du spider de Google est donc devenu beaucoup plus sophistiqué qu'avant. Et cela a de nombreuses conséquences pour le SEO.

Mais il faut bien séparer deux cas :
- Le cas où vous souhaitez qu'un contenu donné soit indexé ;
- Le cas où, au contraire, vous souhaitez empêcher Google de « trouver » un lien, une ressource ou un contenu.

Si vous souhaitez que Google indexe un contenu, les recommandations n'ont pas changé fondamentalement depuis les débuts du moteur en 1998 : ne faites rien qui puisse gêner où empêcher l'exploration des contenus par Googlebot. Et malgré les progrès de celui-ci, il reste déconseillé de générer ces contenus en Javascript…

On sait de plus que si vous voulez qu'un contenu soit bien indexé et se positionne correctement, c'est une bonne idée de s'assurer que le contenu en question soit visible par défaut.

Si par contre, vous souhaitez empêcher Google de trouver un lien, une ressource ou un contenu, alors certaines méthodes ne marchent plus du tout à coup sûr :

- Certains scripts Ajax sont exécutés, et le contenu ainsi généré, exploré ;

- Les « liens » en Javascript sont explorables par Google, quelle que soit la méthode utilisée. Dès lors qu'on est téléporté de la page courante vers une autre page en exécutant un Javascript, Google découvre le « lien » associé. Par contre, Google ne teste pas toutes les interactions possibles sur une page (il ne découvrira pas tous les liens par cette méthode).

Enfin, pour le futur, on peut faire confiance à Google pour améliorer encore sa compréhension des pages, et pour tenir compte du contenu visible sur la version smartphone d'une page, dans l'index du moteur présenté aux utilisateurs de mobiles.

Cela veut donc dire que, dès maintenant, il faut non seulement travailler ses optimisations en regardant tous les contenus figurant dans le code source des pages, mais aussi se préoccuper de ceux qui sont réellement visibles sur la page, lors de la consultation par un internaute. Il va très vite devenir impossible d'optimiser correctement une page pour le SEO, sans se préoccuper de l'impact que cela aura sur l'expérience utilisateur… Mais cela signifie également que l'inverse est vrai, et que de plus en plus, les choix UX vont avoir une influence directe sur le SEO.

## Bibliographie

L'outil « Lynx » pour tester vos pages avec l'œil d'un spider classique :
http://lynx.isc.org/

Le Hangout de John Mueller qui a « révélé » que les contenus non visibles par défauts étaient « écartés » *(discounted)*
https://plus.google.com/u/0/events/cjcubhctfdmckph433d00cro9as

La page du blog webmaster de Google annonçant que Google sait crawler le Javascript *Understanding web pages better* (23 mai 2014)
http://googlewebmastercentral.blogspot.fr/2014/05/understanding-web-pages-better.html

La page du blog webmaster de Google annonçant la mise à jour des guidelines techniques *Updating our technical Webmaster Guidelines* (27 octobre 2014)
http://googlewebmastercentral.blogspot.fr/2014/10/updating-our-technical-webmaster.html

*GET, POST, and safely surfacing more of the web*
http://googlewebmastercentral.blogspot.fr/2011/11/get-post-and-safely-surfacing-more-of.html

**Publications scientifiques**

*VIPS : a Vision Based Segmentation Algorithm*
http://research.microsoft.com/apps/pubs/default.aspx?id=70027

**Philippe YONNET ,** ***Directeur Général de l'agence Search-Foresight, groupe My Media (http://www.search-foresight.com).***