Avant-propos

Avant d'entrer dans le vif du sujet, faisons le tour des domaines abordés par ce livre, ce qu'il contient et ce que j'ai choisi d'omettre. Nous verrons comme le livre articule ses différents thèmes pour en faciliter l'apprentissage.

À qui s'adresse ce livre ?

Toute personne qui s'intéresse de près ou de loin aux technologies web trouvera une utilité à cet ouvrage. Précisons néanmoins qu'*une connaissance préalable des technologies de contenu web statique est préférable* : en l'occurrence, HTML (idéalement XHTML) et CSS.

Il est également *recommandé d'avoir déjà utilisé un langage de programmation* impératif, tel que C, C++, Delphi, Java… Histoire de connaître les notions de variable, fonction, boucle, tableau, etc.

Dans l'idéal, vos connaissances sont à jour, donc conformes aux standards (XHTML strict, CSS 2.1), et solides, notamment en termes de balisage sémantique. Les lecteurs ayant des lacunes sur ces technologies pourront toutefois trouver une présentation succincte des principes fondamentaux dans les annexes A et B, ainsi que de nombreuses ressources — papier ou en ligne— pour parfaire leurs connaissances, dans la bibliographie de ces annexes.

Il n'est par ailleurs pas nécessaire d'avoir des compétences préalables en JavaScript ou DOM, ces sujets étant présentés en détail dans ces pages. En somme, ce livre trouvera son public tant auprès des professionnels chevronnés désireux de se mettre à jour, que des étudiants souhaitant aller au-delà de leurs cours de technologies web, souvent sommaires et trop empiriques, voire obsolètes.

Qu'allez-vous trouver dans ce livre ?

Le livre est découpé en trois parties, précédées de cet avant-propos et d'un chapitre introductif qui présente le Web 2.0 et ses technologies.

Par ailleurs, l'ouvrage est développé sur deux axes forts : un axe thématique et un axe méthodologique et qualitatif. Le premier axe guide le plan, tandis que le second est transversal :

- La **première partie** présente en détail les technologies qui font *vivre* la page web elle-même. Elles sont trop souvent mal connues : JavaScript, DOM et, pour gagner en agilité et en puissance, l'excellente bibliothèque Prototype.
- La **deuxième partie** explore ce qui fait réellement Ajax, à savoir l'objet `XMLHttpRequest`, moteur de requêtes asynchrones, et les frameworks déjà établis dans l'univers du Web 2.0, notamment Prototype et script.aculo.us.
- La **troisième partie** pousse la réflexion et l'utilisation plus loin en ouvrant vos pages sur des contenus et services externes, au travers des Web Services, des API REST et des flux de syndication aux formats RSS et Atom.

L'ouvrage est complété par cinq annexes :

- Les **annexes A et B** fournissent les bases des deux principales technologies de contenu : XHTML et CSS, dans leurs versions récentes.
- L'**annexe C** constitue un plus indéniable : en vous expliquant clairement comment exploiter au mieux les documents de référence qui font le Web (RFC, DTD, recommandations W3C...), elle vous donne les clés d'une connaissance actuelle et faisant autorité.
- L'**annexe D**, elle, est plutôt à lire d'entrée de jeu : elle donne les clés d'un développement plus productif dans votre navigateur, et vous évite de peiner avec les questions de cache en manipulant les exemples de ce livre.
- L'**annexe E** termine avec un tour d'horizon des principaux *frameworks* Java Script, outre Prototype et script.aculo.us, en tentant d'en éliminer rapidement les forces et les faiblesses, et de donner les clés d'un choix pertinent.

Dans tous ces chapitres, j'ai tenté d'insuffler au lecteur le souci constant de la qualité, tant pour la technique elle-même que pour la méthodologie de travail. Qu'il s'agisse d'*unobstrusive JavaScript* (concept que nous étudierons en détail au chapitre 2), de balisage sémantique, de CSS efficaces, d'accessibilité, ou du bon choix de format pour un flux de syndication ou le résultat d'une requête Ajax, j'ai fait de mon mieux pour donner les clés d'un savoir-faire haut de gamme, constituant un avantage compétitif certain, à l'heure où tout un chacun n'hésite pas à clamer sur son CV qu'il est un « développeur web expert ».

Adresse au lecteur

Je me suis posé la question du pronom employé pour désigner l'auteur. D'aucuns diront que le « nous », d'usage, est plus modeste… Peut-être, mais j'écris ce livre pour vous, pour qu'il vous soit utile. Et s'il ne vous plaît pas, c'est *ma* faute, pas *la nôtre*. Alors, pour faire plus simple et plus convivial, ce sera « je ».

Qu'apporte cette deuxième édition ?

Presque deux ans après la première édition de cet ouvrage, il est temps de remettre à jour le contenu, bien sûr, mais aussi d'y effectuer les ajouts et amendements qui font tout le sel d'une nouvelle édition. Au-delà des corrections de coquilles qui avaient échappé à notre vigilance, voici l'essentiel des différences entre la première et la deuxième édition :

- *des clarifications* partout où cela nous a semblé nécessaire, que ce soit lors de nos relectures ou suite à des commentaires envoyés par des lecteurs ;
- *une mise à jour complète du contenu* commandée par l'évolution de l'état de l'art : nouveaux standards, nouveaux outils, nouvelles versions de navigateurs, de bibliothèques (notamment les nombreuses nouveautés de Prototype 1.6) et autres technologies ;
- *un nouveau chapitre crucial :* le **chapitre 5, « Déboguer votre JavaScript »**. Ayant moi-même l'expérience quotidienne de l'écriture de scripts avancés, j'ai constaté que la quasi-totalité des ressources en ligne ou papier étaient lacunaires sur ce sujet. Ce chapitre tente d'apporter une méthodologie et des connaissances techniques pour faciliter la mise au point de scripts JavaScript.
- *un nouveau chapitre* sur les **« Mashups et API 100 % JavaScript »**, qui illustre la tendance désormais répandue d'utiliser directement côté client des services tiers. Nous verrons d'abord l'intégration d'une géolocalisation *via* le célèbre Google Maps, puis avec le tout récent et si sympathique Google Charts.

Les standards du Web

Tout le monde en parle, tout le monde dit que c'est bien et qu'il faut les respecter. Mais personne ne dit vraiment de quoi il s'agit, pourquoi c'est mieux, et vers où se tourner pour mettre le pied à l'étrier.

À l'heure où une portion significative des développeurs web français chevronnés, et la majorité des jeunes diplômés français croyant « connaître le développement web », ignorent ce qu'est le W3C, ne savent pas donner la dernière version de HTML, sont

dans le flou sur les différences exactes entre XHTML et HTML, et pensent que CSS se résume à coller des balises `div` et des attributs `class` et `style` partout, nous avons toujours du chemin à faire en termes d'évangélisation et d'éducation en général.

De quelles technologies parle-t-on ?

Commençons par passer en revue les technologies qui font aujourd'hui figure de standards du Web. Je restreindrai la liste aux technologies qui se rapprochent de notre propos, sous peine d'y consacrer de nombreuses pages.

- **HTML** (*HyperText Markup Language*) est le langage établi de description de contenu dans une page web. Dérivé du SGML, sa syntaxe est un peu trop permissive pour éviter toute ambiguïté et permettre un traitement automatisé vraiment efficace.
- **XML** (*eXtensible Markup Language*) est une syntaxe plus formelle de balisage de contenu, qui garantit le traitement automatique du document sans risquer ni ambiguïtés, ni soucis de jeux de caractères, ni limitations de types ou tailles de contenu.
- **XHTML** revient essentiellement à appliquer à HTML les contraintes syntaxiques de XML, ouvrant ainsi la porte au traitement fiable du contenu des pages web.
- **CSS** (*Cascading Style Sheets*, généralement juste « feuilles de style » en français) est une technologie de présentation permettant une mise en forme extrêmement avancée des contenus compatibles XML (et, par souci de flexibilité, de HTML aussi). Les possibilités sont énormes et vont bien au-delà de ce que permettaient les quelques balises « présentation » de HTML.
- **DOM** (*Document Object Model*) décrit une série d'outils à destination des programmeurs (on parle d'*interfaces*) permettant de représenter et de manipuler en mémoire un document compatible XML. Ces manipulations sont pratiquement illimitées et constituent un des piliers d'une page web « vivante ». Parmi les sous-parties de DOM, on citera notamment *Core*, qui fournit le noyau commun à tous les types de documents ; *HTML*, spécialisé dans les pages web ; et enfin *Events*, qui gouverne le traitement des événements associés aux éléments du document.
- **JavaScript** est un langage de script, dynamique, orienté objet et disposant de nombreuses fonctions avancées, aujourd'hui disponible sous une forme ou sous une autre dans tous les navigateurs un tant soit peu répandus. Sans lui, pas de pages vivantes, pas de Web 2.0, pas d'Ajax !
- **XMLHttpRequest** est un objet capable d'envoyer des requêtes asynchrones *via* HTTP (voilà une phrase qui ne vous dit peut-être pas grand-chose ; pas d'inquiétude, le chapitre 6 vous éclairera bientôt). Utilisé en JavaScript, il constitue le cœur d'Ajax.

- **RSS** 1.0 (*RDF Site Summary*) est un format de flux de syndication, défini de façon beaucoup plus formelle que ses homonymes de versions 0.9x ou 2.0 (où l'abréviation signifie *Really Simple Syndication*), lesquels sont plus répandus mais moins puissants. Il est basé sur **RDF** (*Resource Description Framework*), une grammaire formelle de représentation de la connaissance, autour de laquelle gravite l'univers du Web sémantique (pour plus de détails sur le sujet, consultez par exemple http://www.w3.org/2001/sw/).
- **Atom** est le format de flux de syndication le plus récent, sans doute le plus puissant et le plus efficace aussi, sans pour autant verser dans la complexité.

Parmi les standards du Web, on trouve encore de nombreuses technologies très employées, comme PNG (format d'image), SOAP et les Web Services, XSL et XSLT ; ainsi que d'autres encore trop rarement employées, par exemple SVG (images vectorielles), MathML (formules mathématiques), SMIL (multimédia)...

Qui est à la barre, et où va-t-on ?

Ces standards ne s'inventent pas seuls ; à leur origine, on trouve le plus souvent une organisation, un comité ou une association, parfois une entreprise, plus rarement encore un individu. Mais ceux qui ont fait naître une technologie n'en assurent pas toujours bien l'évolution, comme c'est le cas pour HTML par exemple.

Comprendre qui s'occupe d'un standard permet de savoir où suivre son évolution, de déterminer à quoi s'attendre dans les années à venir, et de mieux comprendre ses orientations et les choix qui les gouvernent.

- **(X)HTML** a été principalement maintenu par le **W3C** (*World Wide Web Consortium*), un groupement international d'associations, d'entreprises et d'individus en charge de la plupart des technologies du Web, principalement dans le contexte des navigateurs. Hélas, après avoir sorti HTML 4.01 en 1999, le W3C a délaissé HTML pour se concentrer sur le Web sémantique, CSS et les technologies autour d'XML.

  Or le problème est que HTML est l'outil fondamental de tout développeur web, quelle que soit la technologie côté serveur, qu'on soit Ajax ou non, Web 2.0 ou non. Figé jusqu'au début du siècle, HTML 4.01 échouait à satisfaire nombre de besoins récurrents.

  Devant la difficulté à remobiliser le W3C autour de HTML, un groupement séparé a vu le jour, le WHAT WG (*Web Hypertext Application Technology Working Group*, http://whatwg.org). Constitué principalement de figures de proue des standards, presque tous par ailleurs membres du W3C, il jouit déjà d'une excellente notoriété et d'une large approbation. Il vise à mettre au point plusieurs standards, dont deux souvent désignés par dérision sous l'appellation commune « HTML 5 » :

Web Applications 1.0 et **Web Forms 2.0**. Ces deux projets augmentent énormément les possibilités pour le développeur web, et plusieurs navigateurs de premier plan ont annoncé leur intention de les prendre en charge...
Pour finir, c'est le W3C qui a repris la main en 2007, et son groupe de travail HTML est reparti sur les bases du HTML 5, développé par le WHAT WG. La prise en charge est prometteuse dans les dernières versions des navigateurs. À surveiller attentivement, donc !

- **CSS** est également l'œuvre du **W3C**, dont il reste un cheval de bataille important. Depuis la version 2, remontant à 1998 (!), le standard évolue de façon double.
  D'un côté, une version 2.1 est en chantier permanent (la dernière révision date de juillet 2007) et constitue une sorte de correction de la version 2, qui en précise les points ambigus, ajoute quelques compléments d'information, etc.
  De l'autre, la version 3 est un chantier proprement pharaonique, à tel point que le standard est découpé en pas moins de 37 modules. Parmi ceux-là, certains font l'objet de beaucoup d'attentions, et sont au stade de la *recommandation candidate* (dernière étape avant l'adoubement au rang de standard), ou de *dernier appel à commentaires*. Il ne s'agit au total que de 11 modules sur 37. Pour les autres, soit le travail n'a carrément pas démarré, soit ils disposent d'une ébauche qui, parfois, stagne pendant des années (le module de gestion des colonnes, pourtant réclamé à corps et à cris par beaucoup, a ainsi gelé entre janvier 2001 et décembre 2005, et sa dernière ébauche remonte à plus d'un an !).
  Enfin, même si CSS a d'ores et déjà révolutionné la création de pages web, on verra qu'il existe un bel écart entre les dernières versions et l'état de l'art dans les navigateurs...
- **DOM** est aussi à mettre au crédit du **W3C**. En DOM, on ne parle pas de versions mais de *niveaux*. Le W3C travaille régulièrement dessus au travers de ses sous-projets : *Core*, *HTML*, *Events*, *Style*, *Views* et *Traversal and Range*.
- **JavaScript** a été inventé en 1995 par **Brendan Eich** pour le navigateur Netscape Navigator 2.0. C'est aujourd'hui l'**ECMA**, un organisme international de spécifications, qui gère son évolution au travers des diverses éditions du standard ECMA-262. Brendan Eich continue à piloter la technologie, *via* les travaux actuels sur ES4, c'est-à-dire JavaScript 2.0, et travaille comme directeur technique de Mozilla.
- **XMLHttpRequest** a été inventé par **Microsoft** pour Internet Explorer (MSIE) 5.0. Depuis 2002, des équivalents ont fait leur apparition dans la plupart des navigateurs, au point qu'un standard **W3C** est en cours de rédaction (depuis plus de deux ans…) pour enfin ouvrir totalement la technologie.
- **RSS** est un sigle qui masque en réalité deux technologies bien distinctes. La première version historique, la 0.90, vient de Netscape (1999). Les versions 0.9x

suivantes et 2.0 sont l'œuvre de Dave Winer tout seul dans son coin, et fournissent une solution simple (et même simpliste) aux besoins les plus courants de la syndication de contenu. Il s'agit de standards gelés, qui n'évolueront plus. La version 1.0 est beaucoup plus puissante, mais aussi plus complexe, car basée sur RDF, donc sur un standard formel lourd réalisé par le **W3C**. Elle est encore, pour l'instant, moins utilisée que ses homonymes.

- **Atom** a été défini, contrairement à RSS, dans la stricte tradition des standards Internet : au moyen d'un forum ouvert de discussion, et encadré dès le début par l'**IETF**, organisme international chargé de la plupart des protocoles Internet (comme HTTP). Il gagne sans cesse en popularité, et constitue très officiellement un standard (ce qu'on appelle une RFC) depuis décembre 2005, sous le numéro 4287 (http://tools.ietf.org/html/rfc4287).

Les principaux acteurs des standards du Web sont donc le W3C et, sans doute de façon moins visible pour l'utilisateur final, l'IETF. On constate néanmoins que le premier est parfois prisonnier de sa propre bureaucratie, au point que des groupes externes reprennent parfois le flambeau, comme cela a été le cas autour de HTML avec le WHAT WG.

Ce panorama ne serait pas complet sans évoquer le WaSP (*Web Standards Project*, http://webstandards.org), véritable coalition d'individus ayant appréhendé tout l'intérêt des standards du Web et la portée de leur application. Ce groupe fut un acteur important de l'arrêt de la « guerre des navigateurs » qui fit rage dans les années 1990, laissant Navigator sur le carreau et faisant entrer MSIE dans la léthargie qu'on lui a longtemps connue.

Mais surtout, il œuvre sans relâche pour rallier toujours plus d'acteurs, notamment les éditeurs commerciaux, à la prise en charge des standards. En collaborant avec Microsoft, mais aussi Adobe et Macromedia (du temps où ils n'avaient pas fusionné), ainsi que de nombreux autres, le WaSP aide à rendre les produits phares du marché plus compatibles avec les standards et l'accessibilité. Vraiment, grâces leur soient rendues ! Sans eux, on en serait encore à devoir annoter la moindre mention technique dans un ouvrage comme celui-ci à coups de « IE seulement », « NN seulement », « non supporté », etc.

Quels sont donc ces avantages extraordinaires qui ont convaincu tant de volontaires de fonder ou rejoindre le WaSP, et de partir en croisade auprès des éditeurs ? Quelles perspectives ensoleillées ont-ils vues ? C'est ce que je vais vous expliquer dans la section suivante.

À quoi servent les standards du Web ?

Encore aujourd'hui, on rencontre trop de personnes qui, lorsqu'on évoque les standards du Web, rétorquent : « *Et alors ? Je n'utilise pas tout ça et mon site marche ! Pourquoi diable devrais-je faire autrement ?* ».

Il s'agit là d'une vue bien étroite. Sans vouloir les vexer, cela revient à ne pas voir plus loin que le bout de son nez, à ne se préoccuper que de soi et de son environnement immédiat, ce qui est pour le moins inattendu s'agissant d'un contenu censé être accessible depuis le monde entier, souvent pour longtemps.

L'expression désormais consacrée « *HTML des années 1990* » est utilisée pour désigner ce mélange d'habitudes techniques aujourd'hui dépassées : balisage hétéroclite mêlant allègrement forme et fond, utilisant à mauvais escient certaines balises (ce qu'on appelle de la *soupe de balises*) ; emploi inapproprié ou incohérent de CSS ; surabondance d'éléments `div` ou d'attributs `class` superflus (syndromes baptisés *divitis* et *classitis*) ; déclinaisons manuelles ou à peine automatisées des pages suivant les navigateurs visés ; et bien d'autres usages, que je ne saurais citer tous ici.

Cette façon de faire, fruit d'une approche fondamentalement empirique du développement web et d'une évolution souvent organique des sites, sans cohérence préalable, était peut-être inévitable pour la première génération du Web. Après tout, la version initiale d'un projet regorge souvent d'horreurs qu'il faut ensuite éliminer. Mais il ne s'agit pas ici que d'esthétique. Les conséquences pénibles de cette approche sont nombreuses, et accablent aujourd'hui encore un grand nombre de projets et de sociétés qui persistent à ne pas évoluer :

- Faute d'une utilisation intelligente de CSS et de JavaScript, les pages sont beaucoup trop lourdes, constituées pour 10 % ou moins de contenu véritable. Impact : le coût de la bande passante pour votre site. Pour un site très visité (à partir du million de visiteurs uniques par mois), le superflu atteint un tel volume que son coût se chiffre fréquemment en dizaines voire en centaines de milliers d'euros par mois.
- Un balisage lourd ou rigide, ainsi qu'un emploi inadapté de CSS, créent des pages s'affichant de diverses façons en fonction du navigateur, sans parler des modes de consultation alternatifs répandus : assistant personnel (PDA), téléphone mobile 3G, borne Internet sans souris dans un espace public, *Tablet PC*, impression papier pour lecture ultérieure, et j'en oublie. Pour toucher une vaste audience, il faut alors en passer par la spécialisation de chaque page, travail ô combien fastidieux aux conséquences néfastes : d'une part il multiplie l'espace disque nécessaire, d'autre part il est généralement traité à la légère, de sorte qu'immanquablement certaines versions des pages seront de piètre qualité, voire pas à jour.

- Une mauvaise utilisation des CSS entraîne généralement une intrusion de l'aspect dans le contenu, et rend l'apparence des pages difficile à modifier globalement. Toute refonte de la charte graphique d'un site devient un cauchemar, à force de devoir dénicher tous les styles en ligne et les balises de mise en forme restées cachées au fond d'une page.
- À moins d'avoir été profondément sensibilisé à la question, une équipe de conception et développement de sites web aura tendance à enfreindre à tour de bras les règles d'or de l'accessibilité. Les pages seront donc difficilement exploitables par les non-voyants, les malvoyants, les personnes souffrant d'un handicap, même léger, rendant inenvisageable l'utilisation de la souris, mais aussi par les programmes de traitement automatique... Sachant que le plus grand internaute aveugle de la planète s'appelle Google, les sites peu accessibles sont par conséquent beaucoup moins bien classés dans les moteurs de recherche que s'ils respectaient les principes fondamentaux d'accessibilité. N'oublions pas que le Web est censé, par définition, être accessible par tous. Dans *World Wide Web*, il y a *World*.

À l'inverse, faire évoluer ses méthodes de travail pour garantir un niveau certain de qualité, pour trouver une façon plus moderne et finalement plus *facile* de réaliser des sites web, cela produit rapidement des bénéfices :

- Un site séparant clairement le contenu (balisage XHTML) de la forme (feuilles CSS) et du comportement (scripts JS, c'est-à-dire JavaScript) produira nécessairement des pages infiniment plus légères, sans parler de l'efficacité accrue des stratégies de cache des navigateurs devant un découpage des données en fichiers distincts, qui deviennent par ailleurs déportables sur des réseaux de distribution de contenu (ou CDN, *Content Delivery Networks*) tels que ceux de Yahoo!, Google ou Akamai, qui garantissent un téléchargement rapide et réduisent donc la charge de vos serveurs. Après avoir refondu complètement son site, ESPN, la principale chaîne de sports aux États-Unis, a augmenté son audience tout en divisant à tel point ses coûts de bande passante que l'économie mensuelle, malgré un tarif extrêmement avantageux, se chiffrait en dizaines de milliers de dollars ! (Pour les détails : http://www.mikeindustries.com/blog/archive/2003/06/espn-interview).
- Une utilisation appropriée des CSS implique qu'on a *une seule page XHTML*. J'insiste : *une seule*. Si vous croyez encore qu'il s'agit d'un mythe, allez donc faire un tour sur le CSS Zen Garden (http://www.csszengarden.com). Lorsque vous aurez essayé une petite vingtaine de thèmes, réalisez que la seule chose qui change, c'est la feuille de style. La page HTML est strictement la même. Il ne s'agit pas seulement d'offrir des thèmes, mais bien de proposer une vue adaptée de la page pour de nombreux usages et périphériques de consultation : page agréable à l'impression, mais aussi sur un petit écran (on pense aux PDA et aux téléphones mobiles), ou avec des modes spéciaux pour les mal-voyants (e.g. contraste fort, fond noir, etc.).

Puisqu'il n'y a qu'une seule page, elle est fatalement à jour, et les versions alternatives sont donc sur un pied d'égalité. Tout en gagnant de l'audience, vous la traitez mieux en garantissant le même contenu pour tous.

- Une prise en compte systématique de l'accessibilité, qui elle non plus n'entrave en rien la page, facilite la vie aux utilisateurs touchés par un handicap quelconque (plus de 20 % des internautes aux États-Unis et en France, selon certaines études). Couplée à l'emploi d'un balisage sémantique, elle signifie aussi que l'indexation de la page par les moteurs de recherche sera de bien meilleure qualité, augmentant votre visibilité et donc vos revenus potentiels.

Mercantilisme... aveugle ?

Le patron qui éructait, lors d'une conférence, « on vend des écrans plasma, on s'en fout des aveugles, ce ne sont pas nos clients ! » s'est vu gratifier d'une réponse cinglante : le mal-voyant voire non-voyant n'en est pas moins internaute, et s'il ne peut offrir à un proche un bel écran acheté sur votre site, il l'achètera ailleurs. La réflexion vaut, évidemment, pour tous les handicaps...

Et les navigateurs, qu'en pensent-t-ils ?

Disons qu'ils sont partagés. D'un côté, on trouve les navigateurs libres accompagnés de quelques navigateurs commerciaux traditionnellement respectueux des standards : Mozilla, Firefox, Camino, Konqueror, Opera et Safari, pour ne citer qu'eux. De l'autre, on trouve un navigateur commercial qui ne s'est réveillé que vers 2006, après une léthargie qui aura duré environ 7 ans, j'ai nommé MSIE.

La situation n'est toutefois pas si claire : tous les navigateurs n'ont pas le même niveau de prise en charge pour tous les standards, et le travail sur MSIE a repris à partir de la version 7, avec des progrès encore bien plus importants côté standards pour la version 8 à venir. Dressons ici un rapide portrait des principaux navigateurs au regard des standards. Gardez à l'esprit que cette situation évolue rapidement, et que vous aurez intérêt à suivre l'actualité des principaux navigateurs pour vous tenir à jour.

Notez que tous les navigateurs ci-après supportent `XMLHttpRequest`. MSIE utilise encore un ActiveX qui deviendra un objet natif JavaScript standard dans IE7, tandis que les autres navigateurs utilisent déjà un objet natif JavaScript.

On le voit, MSIE fait de gros progrès avec sa prochaine version 8, mais reste loin derrière les autres en termes de JavaScript et de DOM. Il ne faut pas s'étonner si Konqueror gagne du terrain chez les utilisateurs de Linux, et si Firefox continue, après 4 ans de vie, à grignoter des parts marché (27 % en France, 34 % en Europe soit près de 150 millions d'internautes, et jusqu'à 46 % dans certains pays). Quiconque continue à développer un site Internet au mépris des standards ferait mieux de ne pas avoir trop d'ambitions commerciales...

Mozilla/Firefox/Camino	
Ils utilisent peu ou prou le même moteur, même si la suite Mozilla a été abandonnée par la Fondation et que sa mise à jour est désormais assurée par une communauté de volontaires, qui peuvent parfois mettre du temps à intégrer les nouveautés de Firefox dans la suite complète (projet Seamonkey). Quant à Camino, c'est un Firefox spécialisé Mac OS X, globalement équivalent côté standards. On parle ici de Firefox 3.	
(X)HTML	Très bon support, à hauteur des versions récentes.
CSS	Excellent support du 2.1, et support de plusieurs modules 3.0.
JavaScript	Naturellement le meilleur, puisque le chef d'orchestre de la technologie travaille pour la Fondation Mozilla. Toujours à jour sur la dernière version. Firefox 3.0 prend en charge JS 1.8, restant ainsi le navigateur le plus avancé sur la question.
DOM	Très bon support du 2, support partiel du 3.

Safari 3.1	
(X)HTML	Très bon support, à hauteur des versions récentes.
CSS	Très bon support du 2.1, hormis quelques aspects encore exotiques (notamment les styles audio).
JavaScript	Bon support du 1.5.
DOM	Très bon support du 2, support partiel du 3.

Opera 9.5	
À noter qu'Opera propose une excellente page pour suivre sa compatibilité aux standards : http://www.opera.com/docs/specs/.	
(X)HTML	Très bon support, à hauteur des versions récentes.
CSS	Excellent support de CSS 2.1.
JavaScript	Prend en charge ECMA 262 3rd, soit JS 1.5.
DOM	Très bon support du niveau 2, bon support du 3.

Konqueror 4	
(X)HTML	Très bon support, à hauteur des versions récentes
CSS	Excellent support de CSS 2.1, support partiel de CSS 3.
JavaScript	Bon support du 1.5.
DOM	Très bon support du 2, support partiel du 3.

Internet Explorer 6	
(X)HTML	Très bon support, à hauteur des versions récentes. Un souci avec les prologues XML, sans grande importance.
CSS	Support CSS 1, très partiel de CSS 2.
JavaScript	Jscript 6.0, pas totalement compatible JavaScript, fonctionnellement entre JS 1.3 et 1.5.
DOM	Support correct du niveau 2, mais persiste à se comporter parfois différemment du standard !

Internet Explorer 7	
(X)HTML	Plus de souci de prologues
CSS	Support quasi total de CSS 1, et assez bon de CSS 2.1.
JavaScript et DOM	Aucune amélioration significative depuis la version 6. Entre le DOM des objets `select`, le `getElementById` qui utilise aussi l'attribut `name`, les prototypes non modifiables des objets natifs, le modèle événementiel propriétaire (notamment pas de `addEventListener`) ou l'absence très pénible de DOM niveau 3 XPath, il y a de quoi faire pour IE8...

Internet Explorer 8 (sur base de la beta)	
(X)HTML	beaux progrès, avec entre autres une implémentation partielle de HTML 5.
CSS	très gros progrès sur CSS 2.1 (passe Acid2), et exploite le mode « conforme aux standards » *par défaut*.
JavaScript et DOM	progrès sur quelques bogues incontournables du DOM et sur le ramasse-miettes, mais rien pour le moment quant à l'API d'événements, ni sur le langage JavaScript lui-même… De vrais outils de débogage cependant, largement inspirés de Firebug !

Quelques mots sur les dernières versions

Voici un rapide tour d'horizon des versions en cours et à venir pour les principaux standards. Là encore, un peu de veille sera votre meilleur atout.

- **HTML est en version 4.01** (décembre 1999), **et la version 5 est en plein chantier**. La prise en charge de cette future version par les navigateurs est déjà prometteuse (que ce soit dans Firefox 3, Opera 9.5, Safari 3.1 ou Internet Explorer 8).
- **XHTML est en version 1.1**. La plupart des navigateurs implémentent au moins la 1.0. La 1.1 est plus stricte et demande normalement un type MIME distinct, qui panique notamment MSIE 6 ! En revanche, certains aspects de la prochaine version de XHTML, la 2.0, sont dénigrés par le plus grand nombre, au motif principal qu'elles cassent vraiment trop la compatibilité descendante sans grand avantage en retour. La spécification est d'ailleurs plus ou moins gelée depuis 2 ans.
- **CSS est en version 2.1**, avec beaucoup de travail autour des **37 modules composant CSS 3.0**. Le calendrier de sortie de ces modules à titre de recommandations s'étalera probablement sur au moins 5 ans... Les dernières versions de tous les navigateurs sont « au taquet » sur CSS, avec évidemment un retard certain pour MSIE 8, retard cependant bien moins critique que par le passé.
- **DOM est au niveau 2, et avance bien au niveau 3**, plusieurs modules étant terminés, dont le *Core*. La plupart des navigateurs s'attaquent fermement à ce nouveau niveau, et même MSIE devrait rattraper un peu son retard prochainement.
- **JavaScript est en version 1.8**, actuellement uniquement pris en charge par Firefox 3.0, mais la disponibilité de bibliothèques Java et C++ toutes prêtes (par ex.

Rhino) facilitent l'intégration par d'autres navigateurs. Les versions 1.7 et 1.8 apportent quelques grandes nouveautés, mais sont encore loin de la 2.0 (dont le nom officiel est « ECMAScript 4e Edition », ou « ES4 » pour faire court), dont la sortie ne cesse d'être reportée, mais qui sera impressionnante !

Qu'est-ce que le « Web 2.0 » ?

Le terme « Web 2.0 », qui a envahi la presse et les sites spécialisés, décrit en réalité deux phénomènes distincts.

D'une part il y a cette évolution profonde des interfaces utilisateur proposées en ligne, qui rattrapent en convivialité et en interactivité celles qu'on trouve sur des applications plus classiques (applications dites « desktop », au sens où elles s'exécutent en local sur la machine de l'utilisateur), ou même sur celles qui équipent des périphériques légers (téléphones mobiles, assistants personnels, etc).

Glisser-déplacer, complétion automatique, création dynamique d'images, personnalisation à la volée de l'interface, exécutions en parallèle : autant de comportements que nous avons pris l'habitude de trouver dans les applications, et qui manquaient cruellement — jusqu'à récemment — aux navigateurs. Ceux-ci étaient réduits à des rôles subalternes, à un sous-ensemble ridiculement étriqué de possibilités bien définies. Et pourtant, les navigateurs ne sont pas plus bêtes que les autres programmes : nous les avons simplement sous-exploités jusqu'ici.

Cette première facette s'est récemment trouvé le nom de « RIA », pour *Rich Internet Application*, un terme initialement apparu chez Macromedia, depuis racheté par Adobe. Le navigateur devient véritablement une plate-forme applicative, comme le montrent des applicatifs comme Google Docs ou GMail. On trouve même désormais d'impressionnantes séries d'outils graphiques avancés (dessin vectoriel, illustration, retouche photo, montage vidéo…) entièrement en ligne. Certains mêmes n'hésitent plus à dire que le système d'exploitation (par exemple Windows®) n'est qu'un « ensemble de pilotes destiné à faire fonctionner le navigateur » !

L'autre facette, qui est la caractéristique principale du Web 2.0, est ce qu'on pourrait appeler *le Web aux mains des internautes*. On est passé d'un contenu créé par des « experts », à un contenu directement livré par l'internaute. Ainsi il n'y a pas si longtemps, consulter une page web constituait une expérience similaire à la lecture d'une page imprimée dans un magazine : on n'avait pas son mot à dire sur l'aspect. Cette barre de navigation sur la droite vous gâche la vue ? Ce bandeau de publicité vous énerve ? Le texte est trop petit, ou le contraste trop faible pour votre vue ? Tant pis pour vous ! Le concepteur graphique du site l'a voulu ainsi, et sa volonté fait loi. En fait, consulter une page web était encore *pire* que lire un magazine : sur ce dernier, au

moins bénéficiait-on de l'excellente résolution de l'impression, de sorte que les textes en petite casse restaient lisibles. Bien sûr, la plupart des navigateurs permettent de désactiver CSS ou d'utiliser une feuille de style personnelle, ou encore de zoomer le texte, voire toute la page (images comprises), mais c'est une piètre consolation.

Et voilà que de nouveaux usages apparaissent, qui donnent enfin à l'internaute la haute main sur l'aspect final de la page *sur son navigateur*. *Exit*, les parties superflues et irritantes ! Agrandi, le texte principal écrit trop petit ! Et tant qu'à faire, augmentons la marge entre les paragraphes et aérons le texte en changeant l'interligne ! À l'aide d'outils dédiés, par exemple les extensions GreaseMonkey (http://greasemonkey.mozdev.org/) et Platypus (http://platypus.mozdev.org/) pour Firefox, le visiteur peut ajuster comme bon lui semble l'aspect d'une page, et rendre ces ajustements automatiques en prévision de ses visites ultérieures.

Par ailleurs, les internautes peuvent maintenant contribuer à l'actualité du Web, tant grâce à la facilité de publication qu'offrent des outils comme les blogs, qu'au travers d'annuaires de pages très dynamiques basés sur des votes de popularité (par exemple Digg ou Reddit). Le principe est simple : ces annuaires permettent à tout un chacun de « voter » pour une page quelconque du Web, afin de dire quelque chose comme « Hé ! J'ai aimé cette page ! ». Les annuaires maintiennent alors une liste, par popularité décroissante, des pages ainsi signalées.

Si un nombre massif d'internautes votent pour une même page, celle-ci apparaît fatalement en excellente position dans l'annuaire qui a recueilli les votes. Le résultat net est séduisant : les hauts de liste pour ces annuaires sont là-haut parce que leur contenu a intéressé, amusé ou marqué un maximum de gens. Statistiquement, il a donc toutes les chances de vous intéresser, vous aussi.

Sous un angle plus politique, cela signifie que les gros titres ne sont plus confiés à un comité de rédaction, si facile à instrumentaliser. Pour acquérir une telle visibilité, fût-elle éphémère, la page n'a d'autre choix que de plaire à beaucoup de monde. C'est un système très démocratique.

Les principaux sites de ce type : del.icio.us (http://del.icio.us/) et Digg (http://www.digg.com/, plus orienté technologie), pour n'en citer que deux, sont déjà extrêmement visités (plusieurs dizaines de millions de visiteurs uniques par jour). Du coup, de nombreux blogs, magazines en ligne et autres sites au contenu très dynamique affichent systématiquement sur leurs pages des liens graphiques aisément reconnaissables pour faciliter (et donc encourager) le vote de l'internaute auprès des principaux annuaires.

Les sites Technorati et del.icio.us figurent également parmi les pionniers d'un nouvel usage qui se répand rapidement : le *tagging*. Il s'agit de permettre aux internautes de qualifier une page à coup de mots-clés, pour obtenir un système riche de références croisées et de catégories tous azimuts, bien plus souple que les hiérarchies de catégories

habituelles. Tous les outils de blog, comme Typo (http://typosphere.org/) ou Dotclear 2 (http://www.dotclear.net/), proposent désormais l'affectation de *tags* (étiquettes) aux billets.

Le Web 2.0, tout comme Firefox à sa sortie, vous invite finalement à « reprendre la main sur le Web ! »

Vue d'ensemble, chapitre par chapitre

Pour finir cet avant-propos (un peu dodu, je vous l'accorde), je vous propose de jeter un coup d'œil général à la structure de l'ouvrage, en soulignant son articulation et le rôle de chaque chapitre.

- **Le chapitre 1, « Pourquoi et comment relever le défi du Web 2.0 ? »** pose la problématique et les enjeux. Il s'agit de bien saisir le saut conceptuel entre les sites classiques et le Web 2.0 ; après de nombreux exemples illustrés et le positionnement des principales technologies dans l'architecture globale d'un développement web, le chapitre démystifie Ajax et termine en dressant un plan d'action, autour de cet ouvrage, pour vous aider à tirer le maximum de votre lecture.

Première partie : donner vie aux pages

Quatre chapitres visent à s'assurer que vous maîtrisez bien les piliers désormais classiques sur lesquels se construit aujourd'hui Ajax. À moins que vous ne soyez véritablement un expert en JavaScript et DOM, parfaitement respectueux des standards qui les gouvernent, je ne saurais trop vous recommander de ne *pas* faire l'impasse sur ces chapitres, au seul prétexte que vous croyez les connaître. Il y a fort à parier que vous allez y apprendre quelque chose.

- **Le chapitre 2, « Ne prenez pas JavaScript pour ce qu'il n'est pas »** présente en détail ce langage si mal connu, accablé d'*a priori* et souvent bien mal employé. Ce chapitre est très riche en conseils et astuces méthodologiques, et prend soin de vous aider à réaliser une couche « comportement » la plus propre et la plus élégante possible.
- **Le chapitre 3, « Manipuler dynamiquement la page avec le DOM »**, nous ouvre les voies royales qui mènent aux pages véritablement dynamiques, dont le contenu évolue rapidement, entièrement côté client. De nombreux exemples pour des besoins concrets sont présentés. Des conseils précieux et un point sur les problèmes résiduels de compatibilité terminent ce chapitre.
- **Le chapitre 4** présente la quasi totalité de **Prototype**, sans doute la plus utile des bibliothèques JavaScript les plus répandues. Grâce à elle, nous allons apprendre à

réaliser du code JavaScript, non seulement plus portable que nos précédents exemples, mais encore plus élégant, plus concis, plus expressif, et d'une façon générale tellement plus agréable à écrire…

- **Le chapitre 5** tente de combler une terrible lacune de la plupart des ressources actuelles, en vous montrant dans le détail comment **déboguer votre JavaScript**, que ce soit sur Firefox, Safari, Opera ou Internet Explorer. Comme dans le reste de l'ouvrage, plus qu'un simple passage en revue des outils, c'est une véritable méthodologie de travail qui sera proposée.

Deuxième partie : Ajax, ou l'art de chuchoter discrètement

Une fois vos bases techniques bien solides et confortables, vous allez pouvoir vous plonger dans ce qui constitue, pour beaucoup, la partie la plus visible d'Ajax : les requêtes asynchrones en arrière-plan. C'est grâce à elles que nos pages semblent enfin capables de « faire plusieurs choses en même temps », et n'ont plus autant besoin de se recharger intégralement.

- **Le chapitre 6, « Les mains dans le cambouis avec XMLHttpRequest »**, vous emmène jusqu'aux tréfonds de la technologie responsable des requêtes asynchrones. C'est l'occasion de découvrir une autre technologie de pointe, très agréable elle aussi : le langage Ruby, dont nous nous servirons pour créer, avec une déconcertante facilité, un serveur web à contenus dynamiques pour nos tests.
- **Le chapitre 7, « Ajax tout en souplesse avec Prototype »**, nous fait passer à la vitesse supérieure ! Puisque nous maîtrisons désormais les rouages, nous allons pouvoir délaisser le cambouis pour faire des bonds spectaculaires en productivité avec les facilités Ajax de Prototype.
- **Le chapitre 8, « Une ergonomie de rêve avec script.aculo.us »** explore l'incroyable bibliothèque d'effets visuels et de comportements avancés proposée par script.aculo.us. Ce chapitre vous emmène par ailleurs plus loin dans la réflexion, autour des usages pertinents ou malvenus d'Ajax et des limites de son utilisation.

Troisième partie : Parler au reste du monde

C'est un peu la partie bonus, qui va au-delà de la technologie Ajax pour explorer des usages concrets et de plus en plus fréquents. L'idée, c'est que nos pages n'ont aucune raison de se limiter à notre serveur, et peuvent discuter tout aussi aisément avec n'importe quel site et n'importe quel service prévu à cet effet.

- **Le chapitre 9, « WebServices et REST : nous ne sommes plus seuls »**, illustre cette idée en présentant ces deux technologies pour s'atteler ensuite à faire profiter nos pages des possibilités de recherche d'Amazon, de prévision de The Weather Channel, et des bibliothèques d'images de Flickr.

- **Le chapitre 10, « L'information à la carte : flux RSS et Atom »**, présente les deux principaux formats de flux pour mettre en œuvre une syndication de contenus (blogs et autres) directement sur nos pages.
- **Le chapitre 11, « Mashups et API 100 % JavaScript »**, illustre la nouvelle tendance à la réutilisation de services externes directement côté client, au travers du célèbre service Google Maps et du plus récent (mais tout aussi sympathique) Google Charts.

Des annexes pour le débutant comme pour l'expert

Sur cinq annexes, deux visent à aider le lecteur auquel manqueraient quelques bases, tandis que les trois dernières donnent à tous des compétences recherchées.

- **L'annexe A, « Bien baliser votre contenu : XHTML sémantique »**, redonne les bases du XHTML et insiste lourdement sur l'importance d'un balisage non seulement valide, mais surtout sémantique. Après avoir succinctement donné la liste des balises pour mémoire, elle fournit également quelques cas concrets de balisage impeccable, correspondant à des besoins récurrents.
- **L'annexe B, « Un aspect irréprochable et flexible : CSS 2.1 »**, joue le même rôle vis-à-vis de CSS, et donc de la mise en forme. Le vocabulaire est précisé, avant d'attaquer suffisamment les fondamentaux : structure des règles, principe de cascade et modèle de boîtes. Une liste concise des sélecteurs et propriétés permet de ne pas trop patauger dans les exemples du reste de l'ouvrage.
- **L'annexe C, « Le plus de l'expert : savoir lire une spécification »**, apporte une réelle plus-value en vous apprenant à lire les principaux formats de spécification pour les standards du Web et à naviguer au sein de ces documents parfois complexes, qui font souvent appel à des syntaxes particulières. Être à l'aise avec ces documents présente de nombreux avantages, et constitue *une compétence encore trop rare*.
- **L'annexe D, « Développer avec son navigateur web »**, fait le point sur les possibilités plus ou moins riches pour votre productivité de développeur web sur les principaux navigateurs : gestion du cache, extensions, outils complémentaires de débogage et de test, autant d'outils qui sont passés en revue. *À lire impérativement, en fait, avant de démarrer le livre !*
- **L'annexe E** enfin, **« Tour d'horizon des autres frameworks JavaScript »**, donne un aperçu des frameworks les plus populaires après Prototype et script.aculo.us, avec notamment jQuery, YUI, Ext JS et Dōjō.

J'ai fait de mon mieux pour que vous retrouviez dans cet ouvrage autant d'informations techniques, concrètes et de qualité, que ce que je fournissais à mes étudiants dans mes cours.

Aller plus loin…

On ne le répétera jamais assez, tout l'ouvrage tente de vous insuffler le constant souci de la qualité, de l'élégance, de l'efficacité, au travers de nombreux conseils méthodologiques et choix techniques savamment orientés. L'objectif n'est rien moins que vous rendre meilleur(e) que vos concurrents !

Dans le même esprit, la plupart des chapitres se terminent par une section « Pour aller plus loin... », qui donne la liste des ouvrages et ressources en ligne permettant d'approfondir les sujets explorés.

À propos des exemples de code

L'ensemble des codes source de ce livre est disponible dans une archive mise en place sur le site web des Éditions Eyrolles, accessible depuis la page de l'ouvrage. Certains chapitres n'utilisent que de courts extraits (par exemple, le chapitre 4 sur Prototype), mais l'archive fournit toujours des pages de test complètes.

Ces exemples ont tous été testés sur Firefox 2, Firefox 3, Safari 3.1, MSIE 6, MSIE 7, MSIE 8b1 Opera 9.5 et Konqueror 3.5.2. Lorsque certaines contraintes sont incontournables, leur impact est précisé dans le texte du livre.

Par ailleurs, les bibliothèques Prototype et script.aculo.us qui y figurent sont parfois plus récentes que leur dernière version stable publique (1.6.0.3 et 1.8.1 respectivement). L'archive de codes source vous fournit aussi ces versions à part, dans un répertoire `bibliotheques_seules` à la racine de l'archive.

Remerciements

Ce livre n'aurait pas vu le jour sans la confiance que m'ont témoignée Muriel Shan Sei Fan et Éric Sulpice. Leur bonne humeur, leur amour de l'informatique et leur dynamisme m'ont d'abord donné envie d'écrire pour Eyrolles, et par la suite grandement facilité la tâche. Un gros merci à Muriel, notamment pour avoir fait sentir très tôt le besoin d'un *extreme makeover* sur la table des matières !

Xavier Borderie et Richard Piacentini ont eu la gentillesse d'assurer la relecture technique de la première édition. Raphaël Goetter, gourou des CSS, a également accepté de relire l'annexe B, en dépit de son planning de ministre. Le livre a énormément bénéficié de leurs apports et remarques constructives. Ce que vous y aimerez, vous le leur devrez certainement. Si certaines parties vous déçoivent, la faute sera mienne. J'adresse également toute ma gratitude à Tristan Nitot pour avoir accepté de rédiger la préface, ce que je considère comme un bel honneur.

Enfin, ma compagne Élodie Jaubert a supporté mon manque de disponibilité pendant les quelque trois mois d'écriture, et m'a soutenu sans faillir avec beaucoup d'amour, au point même d'accepter, au cœur de la tourmente, de devenir ma femme. Ce livre est là, avant tout, grâce à elle.

Et pour cette deuxième édition…

Déjà deux ans… Et 5 000 exemplaires ! Le livre s'est établi une solide petite réputation de référence et les commentaires en ligne et courriels que j'ai reçus vont au-delà de mes espérances.

C'est donc avant tout les lecteurs de la première édition que je souhaite remercier ici, car sans eux, on ne remettrait pas le couvert. J'aime à croire que cette nouvelle mouture a suffisamment de bonnes choses en elle pour se justifier pleinement à leurs yeux !

Élodie, elle, croit toujours en moi et me donne la force d'avancer, d'écrire, de partager. Merci pour tout, mon ange.