Nicolas JEAN

XHTML, CSS : le futur du web a déjà ses standards

De la présentation archaïque et linéaire des premiers sites web à aujourd'hui, il ne s'est écoulé que 10 ans ; pourtant les standards disponibles aujourd'hui ont su tirer rapidement partie de cette courte expérience pour offrir un éventail d'outils et de langage d'une grande richesse.

Pour vos surfs sur le web, je ne saurais que trop vous conseiller d'utiliser l'excellentissime Mozilla / Firebird. Pourtant force est de constater, pour qui possède un site un peu visité, que ce n'est pas le navigateur le plus utilisé. Pire même les sites dédiés à Linux sont majoritairement visités par Internet Explorer. Ce monopole n'a fait qu'en remplacer un autre, celui de Netscape aux premières heures. Ces quasis monopoles ont eu tendance à fausser l'utilisation des langages servant à faire des sites, les webmasters utilisant les fonctionnalités disponibles dans le navigateur vedette plutôt que le standard. Mais l'arrêt du développement de Netscape, et à présent celui de IE (pour un période indéterminée) devrait permettre de retrouver le droit chemin avec le suivi des standards édités par le w3c (world wide web consortium, www.w3c.org). D'ailleurs même si chaque browser essaye de faire son petit lobbying, la majorité des navigateurs de dernière génération supporte une grande partie des standards.

Les langages utilisés pour décrire le contenu d'un site web sont l'HTML (HyperText Markup Language), aujourd'hui dans sa quatrième et dernière version, et les CSS (Cascading Style Sheet). HTML est arrivé au terme de son évolution et doit maintenant laissé la place à XHTML.

De HTML 4.01 à XHTML 1.0

Avec l'apparition de nouveau terminaux pour visualiser des pages web (PDA, téléphone, …) et le besoin incessant de nouvelles fonctionnalités, HTML a fini par montrer ces limites en matière d'évolutivité. Pour pallier cet écueil, un nouveau langage de description a vu le jour XHTML (Extensible HyperText Markup Language). C'est un mixe entre HTML et XML, c'est-à-dire que l'on retrouve toutes les fonctionnalités disponible dans HTML 4.01 mais avec la syntaxe, la rigueur et la souplesse d'évolution fournit par XML. Ces nouveaux documents présentent les avantages suivants :

- Avec un document XHTML conforme au XML, il est possible d'utiliser tous les outils standards disponibles pour éditer et travailler sur le XML.
- Avec un document XHTML conforme au XML, il est plus facile d'écrire des programmes de traitement de données, la présentation et le contenu sont séparés.
- Avec l'évolution du standard, un document en version 1.0 pourra plus facilement interagir avec une version ultérieure.

Pour définir un langage basé sur XML, on utilise une DTD. On retrouve donc naturellement 3 DTD pour XHTML 1.0 (les équivalents de HTML 4) :

- DTD XHTML 1.0 Strict : pour vos pages dans un pur esprit XHTML, sans concession

- DTD XHTML 1.0 Transitional : si vous souhaitez avoir un peu de souplesse et prendre le temps de migrer vos pages (et/ou vos compétences) en douceur
- DTD XHTML 1.0 Frameset : pour les frames

Un Document, avec la DTD Transitional (ne soyons pas trop extrémiste), aura donc la forme suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="fr">
  <head>
    <title>Ma premi&egrave ;re page xhtml 1.0</title>
  </head>
  <body>
    <p>dur dur de faire du x</p>
  </body>
</html>
```

Les différences en pratique

Finalement, pour qui a pris l'habitude de coder proprement ses pages, il y a peu de bouleversement majeur. Les points marquants sont les suivants, et sont du à l'utilisation de XML :

- Le document doit être bien formé (well formed), c'est-à-dire que en théorie, on ne doit plus prendre de liberté avec le standard : pas de tag farfelu, pas de tag imbriqué…
- Majuscule et minuscule sont interprétés différemment, en XHTML tous les tags sont en minuscule.
- Tout tag ouvert doit être fermé. Un doit toujours avoir son . Les tags en une seule partie (c'est-à-dire sans contenu) tel que le br, img ou encore hr, se forment de cette façon :
- Les attributs ont toujours la forme : *nom= ″valeur″*; on oublie donc les attributs solitaire tel que nowrap ou les width=4 et on aura *nowrap= ″nowrap″* et *width=″5″*
- L'attribut name ne doit plus être utilisé (il est obsolète/deprecated). Il servait notamment pour les ancres dans une page. A la place on utilisera l'attribut id, qui servait déjà pour les feuilles de style.
- Les sections style et script (pour les CSS et le Javascript) sont de type #PCDATA, les caractères < et & ne sont donc pas autorisés sous cette forme, le plus simple est donc de faire:

```
<script type="text/javascript">
    <![CDATA[
    ... contenu du script ...
    ]]>
</script>
```

A part ces quelques règles d'écriture il n'y pas de grand bouleversement dans XHTML 1.0 par rapport à HTML 4. Les tags, les attributs et les évènements sont quasi identiques. Il vous suffit dans de faire les quelques ajustement dans l'écriture de vos pages et c'est gagné (en supposant que vos pages étaient déjà au standard HTML, sinon vous avez du boulot).

Les standards c'est bien …

Ce nouveau standard, son support (déjà presque) complet dans les nouveaux navigateurs et surtout son usage chez les webmasters petits et grands devrait permettre d'obtenir des sources d'informations sur l'Internet plus homogènes, plus faciles à lire et plus accessibles par programme. En attendant ce jour proche, certains utilisent encore d'anciens modèles de navigateurs, déjà privés de la souplesse et de la rapidité des browsers récents on ne va pas en plus les priver de saines lectures, il faut donc gérer la transition…

… être lu c'est mieux.

Pour ça, il suffit de faire quelques ajustements dans le codage de vos pages xhtml pour qu'elles restent au standard, avec quelques mots clefs obsolètes (*deprecated)*, et restent lisibles pour les navigateurs html pur (et sont très tolérants).

- Les éléments vides tel que <*br*> en html deviendront <*br* /> ; notez l'espace avant le /. De plus l'usage des éléments vides doit être réservé à ceux supportés par l'HTML, vous ne pouvez donc pas remplacer <*b*> <*/b*> par <*b* /> (vu l'intérêt, ç ane devrait pas représenter un gros effort)
- Evitez l'utilisation des feuilles de styles et des scripts en ligne mais préférez l'utilisation des fichiers externes
- Si vous utilisez l'attribut *name*, utilisez aussi l'atttribut *id* avec la même valeur : <*a id="ancre" name="ancre"* >…
- Des générations de scripteurs CGI ont transmis une sale habitude à leurs successeurs PHP et autres systèmes de script (à moins que ce soit les mêmes), celle d'utiliser le caractères & pour séparé les variables dans une URL. Hors ce caractère n'est pas autorisé, il faudrait donc le remplacer par *&* , si possible.

Je n'ai pas listé ici toutes les astuces à mettre en œuvre pour assurer la compatibilité, si vous expérimentez quelques comportements étranges, rendez vous sur l'annexe C de la recommandations XHTML 1.0 pour plus de détails.

De 1.0 à 1.1

Le standard XHTML 1.0 n'est en fait qu'un standard de passage, une transition, une version bêta en quelque sorte du standard XHTML tel qu'il se doit. Les choses sérieuses commencent vraiment avec XHTML 1.1. Cette fois plus de concession, on entre dans le cœur du sujet, fini la migration, plus d'éléments et d'attributs obsolètes (*deprecated)*, un nouveau standard tout beau tout neuf vient de naître.

XHTML Basic est un ensemble de modules qui représente la base commune supportée par toutes les applications susceptibles d'être clientes web. Ensuite viennent se greffer un ensemble de module qui permette d'apporter de nouvelles fonctionnalités ou d'étendre celle existantes. Un

client PDA devra donc supporter le standard Basic mais pas forcément d'autre module qui pourront servir sur un PC.

Les modules de base de XHTML sont :

- **Structure Module** : body, head, html, title
- **Text Module** : abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var
- **Hypertext Module** : a
- **List Module** : dl, dt, dd, ol, ul, li
- **Basic Forms Module** : form, input, label, select, option, textarea
- **Basic Tables Module** : caption, table, td, th, tr
- **Image Module** : img
- **Object Module** : object, param
- **Metainformation Module** : meta
- **Link Module** : link
- **Base Module** : base

On retrouve sans surprise les tags familiers du html amputés de ceux qui ne servent qu'à la présentation, tel que *font*, dont les effets devront être réalisés à l'aide de CSS.

XHTML 1.1 est basé sur XHTML Basic et est le standard destiné à la création de site. On retrouve donc les modules de bases et quelques autres indispensables comme le module de script (généralement pour le javascript), le module de style (pour les CSS) ou encore le module 'Client Side Image Map' pour créer des liens sur des zones d'une image. La volonté de réellement séparer contenu et présentation est très marquée puisqu'il subsiste également un module pour l'attribut style (qui permet de définir un style en ligne) mais il est déclaré obsolète ; de sorte que pour XHTML 2.0, toutes les informations de style devront être dans une CSS.

XHTML un vrai langage

Avec les navigateurs qui existent aujourd'hui les éditeurs de site web ont pris la mauvaise habitude de prendre de nombreuses libertés avec le standard. Les browsers étant suffisamment permissifs pour accepter de nombreuses erreurs et afficher correctement une page. Il est temps de passer aux choses sérieuses et de considérer le XHTML comme un vrai langage de programmation et de s'astreindre à la même rigueur (on à jamais vu un compilateur C ajouter automatiquement des ; dans un code parce que cela semblait logique). Pour ce faire, un seul choix : VALIDATOR !!!

Cet outil, mis à disposition par le w3c, est disponible en ligne au travers d'un formulaire. Vous n'avez qu'à saisir l'url de la page à valider pour obtenir le résultat (pas toujours très explicite, un peu comme un compilo quoi !), mais au moins vous aurez une idée de l'ampleur des dégâts (à noter que validator valide aussi vos pages html, ne vous en privez pas). Lorsque votre site est un peu important, il devient difficile, voire très ennuyeux, de saisir toutes les pages une à une. Wget vient à notre rescousse pour permettre les tests automatiques à travers un petit script.

Vérificateur succinct

```
#!/bin/bash

# extension des fichiers
EXT="xhtml"
# nom du serveur
SITE="www.xhtml_rulez.com"
# goodies former l'url pour validator
SLASH="%2F"
HTTP="http://validator.w3.org/check?uri=http%3A%2F%2F"${SITE}${SLASH}
# PASBON
PASBON=""

if test $# -gt 0; then
        WDIR="${1}/"
        RDIR="${1}${SLASH}"
else
        WDIR=""
        RDIR=""
fi

LS=$WDIR"*."$EXT
FILES=$(\ls $LS)

for i in $FILES; do
        wget -O IZGOOD "${HTTP}${RDIR}${i##*/}" 2>/dev/null
        grep "This Page Is Valid" IZGOOD >/dev/null
        if test $? -eq 0 ; then
                echo "OK "$i
        else
                echo "KO "$i
                PASBON=$PASBON" "$i
        fi
        rm IZGOOD
done

if test ! "$PASBON" = "" ; then
        echo
        echo "Listes des fichiers invalides:"
        echo $PASBON
fi
```

Ce script permet d'appeler automatiquement validator pour qu'il teste vos pages. Une fois que vous avez mis ce script dans un fichier, configurez EXT qui correspond à l'extension de vos page, et SITE le nom de votre serveur. Comme je dispose de mon site en local, je me place à la

racine et je n'ai plus qu'à lancer ce script pour qu'il interroge validator pour chaque page. Pour éviter de faire la vérification sur 500 pages d'un coup, vous pouvez passer le nom d'un répertoire en paramètre, le script ne vérifie que le répertoire donné (il ne supporte d'ailleurs qu'un niveau d'arborescence). Il est assez simple d'étendre ce code pour s'adapter à votre site, ensuite il faut s'armer d'un peu de patience pour obtenir le résultat complet.

Validator permet aussi de tester un fichier en local en envoyant son contenu dans un POST http. Pratique, si vos pages sont complètement statiques ou que vous avez un serveur en local pour effectuer la génération de la page au préalable.

HTML Tidy

HTML Tidy (http://tidy.sourceforge.net) est votre correcteur automatique. Vous lui donnez une page (x)html que vous avez écrit un soir de grosse fatigue et il va se charger (ou du moins faire pour le mieux) de mettre votre code au standard. Il va donc fermer les tags oubliés, ajouter les / manquants etc. Mais ce qui est plus intéressant c'est qu'il peut également traduire votre page html en xhtml, et tout cela en tenant compte des astuces nécessaires pour qu'elle reste compréhensible par le plus grand nombre de browsers (surtout les illettrés en xhtml).

HTML Tidy regorge également d'options indispensables pour le webmaster comme la suppression des espaces, l'indentation ou encore le nettoyage des tags inutiles. Le w3c mets à votre disposition une version en ligne limitée en terme d'option : http://cgi.w3.org/cgi-bin/tidy.

Les CSS

Avec un document au format XHTML, surtout en 1.1, à peu prés tout ce qui servait à mettre en forme et colorer vos documents a disparu. A présent toute la décoration d'un site se trouve dans la section ou le fichier CSS. On a finalement disjoint contenu et forme, la taille des pages devrait diminuer et les traitements sur les fichiers simplifié puisque il y aura beaucoup moins de tri a faire entre contenu et mise en forme.

CSS 2

Vous pouvez vous reporter au n°51 pour une première introduction aux CSS.

Les CSS 2 sont déjà partiellement (mais jamais complètement) supportées par les navigateurs récents, on peut néanmoins les utiliser sans trop d'inquiétude, seule une utilisation très poussée des possibilités de cette recommandation posera problèmes.

Pour lier votre page à une CSS, la meilleure méthode est d'ajouter :
<**link rel=**"stylesheet" **href=**"style.css" **type=**"text/css">

De cette manière le navigateur chargera votre feuille de style sur la première page de votre site et la gardera pour toutes les autres pages. Il y a un attribut qui est souvent négligé : media. En spécifiant le media vous demandez au navigateur de charger un feuille de style différente en fonction du mode d'affichage, on peut donc avoir :

<**link rel=**"stylesheet" **href=**"style.css" **type=**"text/css" **media**=”screen”>
<**link rel=**"stylesheet" **href=**"print_style.css" **type=**"text/css" **media**=”print”>

La feuille de style ne sera pas la même pour l’écran ou pour l’impression. On voit souvent sur des sites un petit icône permettant d’imprimer les informations présentes sur la page, épurée des menus et autres fioritures. Grâce aux CSS 2 ce travail devient complètement inutile, il vous suffit de définir une deuxième feuille de style où vous déclarez tous les tags, classes et ids qui vous souhaitez voire disparaître avec l’attribut :
visibility : hidden

Par exemple :
.menu { visibility : hidden ; }

- La localisation a été largement améliorée, et pour vous le prouver je vous propose d’essayer (munissez vous d’un miroir pour lire le résultat):

direction: rtl; unicode-bidi: bidi-override;

- Il est possible de spécifier la hauteur et la largeur minimale/maximale d’un élément, cela devrait simplifier le casse-tête de certaine mise en page et éviter l’usage des tableaux et des images transparentes pour forcer la taille minimale d’une cellule (ou mieux d’un *div*). On dispose de min-height, max-height, min-width, max-width. Ils s’utilisent comme width et height.

- Les pseudos classes *before* et *after* permettent de modifier la présentation précedent/suivant un élément, mais surtout grâce à la propriété *content* de pouvoir ajouter un texte.

  p.code :before { content : ″// début du programme :″;}
  p.code :after { content : ″// A vos compilos″;}

  Au delà de la présentation il est donc possible de procéder à quelques modifications chirurgicale du contenu. Malheureusement ou heureusement, selon son point de vue, il n’est pas possible d’inclure des tags (x)html dans la valeur de *content*, ce qui offrirait la possibilité de gérer des templates de présentation sans avoir recours à des pages dynamiques. Cette propriété est quand même un paradoxe si l’on retient l’usage des CSS comme une technique de séparation entre le contenu et la forme.

- La pseudo classe *:hover* bien connu de tous pour le changement de style d’un lien texte, est disponible sur tous les tags. Fini les bouts de code javascript hasardeux sur l’évènement onMouseOver, on peut faire les changements dans sa feuille de style, admirer l’élégance de cet exemple :
  td { background-color : #F0F0FF ; }
  td :hover {background-color : white; }

  A l’instar de :hover pour la souris, :focus permet de changer les propriétés en fonction de l’accessibilité clavier.

- Les selecteurs disponibles sont nettement plus nombreux. On retrouve des possibilités proches de XPath. Ainsi :

  input[type= ″radio″] {}

  ne selectionne que les tags input de type radio. Cela permet de ne pas multiplié les noms de classes ou d'id. Avec cet exemple on pourra éviter de changer la propriété border des boutons radio, ce qui donne un rendu bien plus agréable.

- Alors que Internet Explorer offre la possibilité de modifier l'apparence de la scrollbar pour l'assortir au design et couleurs de son site. Les feuilles de style évoluent pour permettre d'assortir votre site à l'environnement de votre utilisateur (beaucoup plus naturel a priori). Avec :

  body { background-color : Background; }

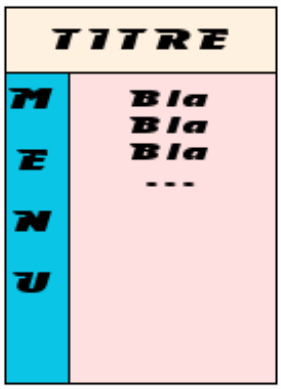
  la couleur de fond de votre site sera la même que la couleur de fond du bureau de l'utilisateur.

- L'utilisation des iframes présentait l'avantage de pouvoir intégrer une frame de façon beaucoup plus souple dans un site web mais en contrepartie le référencement dans les moteurs de recherches devenait impossible. Une alternative est apparut avec l'utilisation de *overflow : auto*. Le résultat de :

  ```
  <div
      style=" height : 200px ; width : 200px ; overflow : auto ;">
      <h1>bienvenue,<br><br>dans mon iframe à base de CSS<br><br></h1>
  </div>
  ```

  sera le même qu'une iframe, c'est-à-dire une boite avec scrollbar de taille 200 par 200 pixels. Malheureusement on se retrouve avec le problème inverse de l'exemple précédent, il n'y a aucun moyen de modifier la couleur de cette barre de défilement qui fait un peu tâche au milieu de notre magnifique design arc en ciel.

- table out, div in
  Pour faire la mise en page d'un site web, et obtenir un bon résultat, l'utilisation des tables s'est généralisée. Pourtant l'objectif des tables est de présenter des résultats ou des listes de données et non de faire le découpage de sa page pour faire un design. De plus pour les utilisateurs de navigateurs non graphiques (lynx, aveugles…), ce format peut être vraiment détestable. Pour ces raisons, et sans doute bien d'autres, il est à présent tout à fait possible de faire le design de son site en utilisant uniquement le tag *div*. Imaginons un site dont la présentation assez simple serait la suivante :



Avec les tables ce serait trivial, voici avec les divs :

```
...
<html>
 <head>
  ...
  <style>
   body { text-align: center; }
   #global {
    text-align: left;
    width: 750px;
    margin-right:auto;
    margin-left:auto;
    border: black 1px solid;
    background: #D0D0FF;
    }

   #entete{
    width: 750px;
    background: #FFFFD0;
    border-bottom: black 1px solid;
    }
   #corps{
    width: 600px;
    margin-left:150px;
    background: #FFD0D0;
    border-left: black 1px solid;
    }
   #menu{
    width: 150px;
    background: #D0D0FF;
    position: absolute;
    }
  </style>
 </head>
```

```
  <body>
   <div id="global">
    <div id="entete"><h1 style="margin: 0px;">Title</h1></div>
    <div id="menu">m<br>e<br>n<br>u<br></div>
    <div id="corps">blablablabalbal<br>e<br>n<br>u<br><br>n<br></div>
   </div>
  </body>
</html>
```

CSS 3

Cette recommandation est loin d'être prête mais si vous aimez faire des sites web, je vous promet beaucoup d'excitation et d'impatience à la lecture des documents de travail en cours de rédaction par le w3c. A l'instar de xhtml 1.1, elle est composée de plusieurs modules. Il faut dire qu'aujourd'hui, on se trouve souvent limité par les capacités de présentation de (x)html, pour pallier nos frustrations, nous avons recourt aux images. Bien que l'on obtienne le rendu désiré, la page devient beaucoup plus lourde à télécharger et difficile à référencer. Ainsi de nombreuses innovations de CSS3 permettront de se passer des images ou encore des tables pour la présentation.

En vrac, parce qu'il est encore bien tôt pour en dire plus et approfondir, on pourra gérer plus finement les couleurs avec des effets de transparence, avoir une bordure en forme de vague ou avec des coins arrondis, faire *scroller* automatiquement un texte dans une boite (sympa pour des news défilantes), faire une mise en page multi colonnes (les CSS pour l'imprimante auront une raison supplémentaires d'exister), traiter différemment la première lettre d'un paragraphe (un peu comme certaine publication papier dont la première lettre est beaucoup plus grande), faire un effet d'ombre sur un texte, plus de souplesse pour l'alignement du texte…

C'est encore un peu tôt donc, surtout lorsque l'on voit l'implémentation actuelle de CSS2 dans les navigateurs. Mais si vous parcourez un peu les documents de travail, vous y trouverez de quoi vous faire envie ☺

Sans paresse, vérifie tes CSS

De même qu'une page xhtml se valide, vous pouvez faire de même avec votre feuille de style, le w3c fournit encore un outil pour cela à l'adresse suivante :
http://jigsaw.w3.org/css-validator/validator-uri.html

Le *css validator* peut être téléchargé sur le site pour une installation locale. Mais comme en général une fois la feuille de style mise au point on n'y touche rarement, cela n'en vaut sans doute pas la peine pour le moment. Surtout que les retours que l'on peut lire sur les personnes qui ont tenté l'expérience n'est pas des plus favorables, beaucoup de sueur en perspective… (peut être un petit guide dans un prochain LM, pour un installation sans peine)

Le futur des navigateurs

La tendance actuelle est d'avoir un 'renderer' pour afficher le contenu des pages d'un site web et de pouvoir l'utiliser à l'infini en l'habillant un peu pour offrir les fonctionnalités de base d'un navigateur, voire beaucoup plus. Les deux renderers principaux sont Gecko et Internet Explorer, et un sérieux outsider pour la qualité du rendu est KHTML.

Internet Explorer

Le prochain slogan de Gecko pourra être : « Internet Explorer le navigateur qui m'fait plus peur » Une interview annonçait la fin du développement de IE avec la version 6.0SP1, avant l'intégration (encore) dans l'OS. Depuis, après un démenti, on ne peut que attendre et voir… IE supporte xhtml 1.0 et entièrement les CSS1 (l'exemple des div ci-dessus ne fonctionne pas).

Gecko

Sans conteste le plus avancé des renderers pour le web, il est magnifiquement (si, si je suis objectif et impartial) intégré dans Firebird. Il supporte lui aussi Xhtml 1.0, CSS1 et partiellement CSS2.

KHTML

Intégré à Konqueror, il supporte HTML 4.0, CSS1 et presque intégralement CSS2. On peut espérer que son développement continue comme ça, surtout depuis qu'il est utilisé dans Safari le browser par défaut disponible sur Mac OS X.

Ces trois renderers ne sont néanmoins pas les seuls et peut-être verrons nous l'émergence d'un autre produit, tel que celui de Opera, qui produit aussi un navigateur multi plateforme, ou même de Netclue qui propose un renderer Java. Ces offres devront néanmoins surmonter l'énorme handicap d'être payant.

Conclusion

Il est peut être encore un peu tôt pour utiliser toutes ces technologies si vous avez déjà un site avec des milliers de visiteurs par jour. La mise en production demande toujours un peu de patience. Par contre si vous décidez de vous lancer dans la création d'un site pour le publier sur l'Internet, c'est une très bonne idée de prendre un browser récent et de vous atteler à la tâche en utilisant XHTML et CSS, ainsi le temps que votre audience soit confortable, vos lecteurs auront un navigateur optimal et vous pourrez profiter des dernières innovations sans surcharge de travail. Votre site sera se distinguer par sa qualité, l'élégance de sa présentation et sa simplicité d'accès. Bon site…

Post Scriptum

J'ai pris intentionnellement le parti du standard, parce que je suis un codeur (j'aime bien les choses claires et définies) et webmaster (je me cogne du *tml, tous les jours), parce que le temps où il fallait faire deux fois la mise en page de son site à cause de Netscape ou IE m'a vraiment

saoulé. Malheureusement, l'approche XHTML et CSS est nettement moins triviale que HTML, avec ce dernier il était simple de commencer un petit site et de faire évoluer la présentation au fur et à mesure de ses besoins en apprenant quelques tags supplémentaires. XHTML, CSS demande nettement plus d'abstraction et déjà une expérience dans la conception et le design de site web pour comprendre la beauté et l'efficacité des options offertes. On peut donc sans grand risque imaginer encore de beaux jours pour le support de HTML dans les navigateurs.

J'ai de plus parlé de standards, là où il n'y a que des recommandations, je vous propose de prendre cela comme un lapsus révélateur.

Nicolas JEAN
http://www.nicolasjean.com

Salemioche.net : création de site web pour les débutants
Nikozen : hébergement professionnel – création site internet
Glaces.org : recettes de glaces et sorbets
Shopping Relax : guide achat en ligne
IP relax : protocole http, smtp, pop, imap, irc, ftp, mime…

REFERENCES

http://www.w3c.org : world wide web consortium, source des standards html, xhtml et css et des outils validator.
http://www.mozilla.org : pour le moteur Gecko et Firebird.
http://www.konqueror.org: Khtml
http://www.opera.com : Opera
http://www.netcluesoft.com: renderer de netclue
http://tidy.sourceforge.net : HTML Tidy