Passer du HTML au XHTML

Par Denis Boudreau et Laurent Jouanneau

Introduction

Depuis le 26 janvier 2000, le XHTML est la nouvelle norme du W3C en matière de langage balisé pour concevoir des documents Web. Que vos pages existantes soient actuellement conformes ou non aux différentes versions du HTML importe peu. Vous allez rapidement constater que les convertir en XHTML n 'est pas sorcier du tout. En effet, puisque le XHTML n'est rien de plus que du HTML reformulé de façon à respecter les règles strictes du XML, il ne vous suffit que d'apprendre quelques règles syntaxiques propres à XML pour commencer à coder selon les normes du W3C. Si vos documents sont déjà conformes (valides) aux règles du HTML 4.01, votre travail de conversion en sera grandement facilité. Cependant, si vous avez l'habitude de coder selon les règles plus permissives du HTML des premières versions, truffé de balises propriétaires ou dépréciées depuis, vous aurez un peu plus de pain sur la planche.

Tout ce qui vous sépare de votre but, c'est un peu moins d'une dizaine de petites lois et quelques principes d'application. Issues de la spécification XML, ces lois permettent une séparation logique entre les aspects de structure et de présentation dans un document Web. Car voilà réellement ce qu'est le XHTML : un pont entre le HTML (le langage d'hier) et le XML (le langage de demain). Cet article n'a qu'un seul but, celui de vous aider à franchir ce pas entre le HTML du siècle dernier et celui d'aujourd'hui ou, si vous préférez, à mettre un peu de *X* dans votre HTML.

Chaque balise nécessite une fermeture...

Dans les premières heures du HTML, on pouvait se permettre d'être relativement brouillon dans la façon d'organiser son code. Maintenant, selon les règles plus strictes du XML, il n'y a plus de place pour une telle permissivité ; ainsi, toutes les balises présentes dans un document Web doivent dorénavant être correctement fermées : il ne faut jamais oublier d'ajouter la balise de fermeture d'un élément quand celle-ci existe :

Invalide : `<p>Lorem ipsum dolor sit amet. Praesent vel justo.`
Valide : `<p>Lorem ipsum dolor sit amet. Praesent vel justo.</p>`

Article original publié sur CYBERcodeur.net et OpenWeb le 21 mars 2003 :

- http://www.cybercodeur.net/weblog/articles/art_20030321.php
- http://www.openweb.eu.org/articles/html_au_xhtml/

... Même celles qui n'en ont pas

En revanche, comment peut-on fermer ces autres éléments ne possédant pas de balise de fin, comme les
 et <img> ? En leur inventant une balise de fermeture ? Eh bien presque. Sauf que vous n'aurez pas à les inventer puisque le W3C s'en est déjà chargé pour vous. Vous pourriez effectivement vous mettre à coder des
</br> ou des <img></img>, mais une telle pratique est déconseillée puisqu'il est possible que cela produise des résultats inattendus dans certains agents utilisateurs. Ce serait aussi un peu inutile puisque selon la syntaxe XML, il est possible de simplement fermer un élément en lui attribuant une barre oblique (un *slash*) en fin de balise, comme ceci :
, ou encore <img />. Cependant, si vous optez pour la seconde méthode, il ne faut pas oublier d'inclure un espace entre le contenu de l'élément et la barre oblique, car autrement, les anciens navigateurs, en particulier Netscape 4.x, ne pourront l'interpréter et l'ignoreront tout bonnement :

Invalide : `<br>`
Valide : `<br />`

Imbriquer correctement les éléments

Quand on ouvre une série de balises en cascades, (les unes à l'intérieur de l'espace de définition des autres), il faut obligatoirement les refermer dans l'ordre inverse de l'ordre d'ouverture pour respecter la structure logique interne du document. Il faut toujours voir une balise HTML comme étant incluse dans une autre balise qui lui sert de parent. Ainsi, dans l'exemple ci-dessous, l'élément , qui est un enfant direct de l'élément <p>, doit impérativement se refermer à l'intérieur de l'élément qui le contient :

Invalide : `<p>Paragraphe avec texte en <strong>gras</p></strong>`
Valide : `<p>Paragraphe avec texte en <strong>gras</strong></p>`

Utiliser des minuscules dans les balises et leurs attributs

En HTML, on pouvait à loisir utiliser des majuscules ou des minuscules pour baliser nos documents. Certains voyaient même en l'utilisation des majuscules un moyen efficace pour repérer le code HTML du contenu dans un document. Avec XHTML, ce n'est plus possible ; puisque XML est sensible à la casse, toutes les balises et tous leurs attributs doivent obligatoirement être écrits en lettres minuscules. C'est donc dire que les balises <LI> et <li> ne sont plus identiques en XHTML, alors que c'était le cas en HTML. Les valeurs d'attributs, par contre, peuvent toujours être écrites en majuscules :

Invalide : `<TEXTAREA ID="monTexte"></TEXTAREA>`
Valide : `<textarea id="monTexte"></textarea>`

Article original publié sur CYBERcodeur.net et OpenWeb le 21 mars 2003 :

- http://www.cybercodeur.net/weblog/articles/art_20030321.php
- http://www.openweb.eu.org/articles/html_au_xhtml/

Chaque valeur d'attribut doit être entre guillemets

Avec les versions antécédentes de HTML, le recours aux guillemets pour encadrer les valeurs d'attributs était conseillé, mais pas obligatoire. Toujours selon les règles de XML, l'utilisation des guillemets n'est plus une proposition, mais bien une obligation. De plus, il ne peut plus y avoir de saut de ligne dans la définition d'une valeur donnée. Les caractères d'espacement (comme les espaces, les bris de lignes et les retours de chariots) sont interprétés différemment de navigateurs en navigateurs. Lorsque des caractères d'espacement sont insérés dans les valeurs d'attributs, les navigateurs tronquent ces espaces et les transposent en code ASCII, d'où parfois d'imprévisibles maux de têtes. Afin de vous éviter ces tracas, prenez simplement la bonne habitude de ne pas laisser d'espace dans vos valeurs d'attributs :

Invalide : `<div id=mon Menu></div>`
Valide : `<div id="monMenu"></div>`

Les formes abrégées d'attributs sont interdites

Certaines balises en HTML possédaient des attributs autonomes qui pouvaient être utilisés sans valeurs associées, comme c'était par exemple le cas pour la balise <input> avec laquelle on pouvait utiliser les attributs checked, disabled et readonly. Une pratique courante en HTML consistait donc à les déclarer de manière abrégée dans le code, afin d'économiser le nombre de caractères, en spécifiant directement un attribut sans valeur associée dans une balise HTML. En XHTML, cette pratique est révolue. Dorénavant, afin de rendre votre code valide, il vous faut l'inscrire de manière complète, c'est-à-dire en spécifiant l'attribut et sa valeur associée, même si cela représente une répétition :

Invalide : `<option value="page.html" selected></option>`
Valide : `<option value="page.html" selected="selected"></option>`

L'attribut name est remplacé par l'attribut id

L'attribut name, utilisé en HTML pour nommer les ancres, les images ou tout autre objet dans un document Web est remplacé par l'attribut id en XHTML. En effet, puisque le principe de nommer un objet revient à l'identifier et que par définition, cet identifiant se doit d'être unique, le recours à l'attribut id permet de s'assurer que la communication par le DOM avec un objet dans un document donné se fera individuellement. Malheureusement, le support pour l'attribut id étant faible ou inexistant dans les anciens navigateurs, il importe (en XHTML 1.0) de continuer à utiliser à la fois les attributs name et id pour désigner un même objet dans ces navigateurs, en leur attribuant des valeurs identiques, de sorte que les navigateurs de nouvelle génération puissent y trouver leur compte conformément aux règles de XML, tout en assurant une rétro compatibilité avec les anciens navigateurs :

Article original publié sur CYBERcodeur.net et OpenWeb le 21 mars 2003 :

- http://www.cybercodeur.net/weblog/articles/art_20030321.php
- http://www.openweb.eu.org/articles/html_au_xhtml/

```
Invalide : <h2 name="titre">...</h2>
Valide : <h2 name="titre" id="titre">...</h2>
```

Ajoutons que l'attribut name des éléments a, applet, form, frame, iframe, img, et map est déprécié et ne peut plus être employé en XHTML dans sa version stricte.

Gestion des caractères spéciaux avec CDATA

XHTML est beaucoup plus sensible que ne l'était HTML aux caractères spéciaux dans les déclarations CSS et JavaScript. Vous ne pouvez plus inclure les blocs de code dans des balises de commentaires comme en HTML : en effet, les navigateurs supportant XML peuvent réagir de manière inattendue à la présence de ces caractères et simplement les ignorer, affichant ainsi le contenu des éléments script et style. Afin d'éviter un tel désastre, il est recommandé d'entourer les scripts et les styles d'une section CDATA, qui indiquera aux navigateurs XML que les caractères spéciaux inclus doivent être interprétés normalement.

Toutefois, cela ne règle que partiellement le problème puisque les navigateurs HTML ignorent le contenu d'une balise XML CDATA et requierent l'utilisation traditionnelle des commentaires HTML. La seule solution viable à ce jour consiste donc à placer toutes les définitions de CSS ou de JavaScript dans des fichiers externes.

Gestion des caractères spéciaux dans les URL

Les caractères spéciaux présents dans les valeurs d'attributs s'avèrent également catastrophiques en XHTML. Afin de contrer le problème, vous devez nécessairement les encoder afin d'éviter que le navigateur ne les interprète de façon erronée. Ainsi, pour tous les caractères spéciaux comme "<", ">" ou "&" destinés à être interprétés tels quels, vous devrez plutôt inscrire "<", ">" ou "&" :

```
Invalide : <a href="index.php" title="Articles & Didacticiels">
Valide : <a href="index.php" title="Articles &amp;amp; Didacticiels">
```

Conformité et type de document

Afin d'être conforme aux normes XHTML, outre les spécificités de syntaxe XML décrites précédemment, un document doit respecter les règles suivantes :

Article original publié sur CYBERcodeur.net et OpenWeb le 21 mars 2003 :

- http://www.cybercodeur.net/weblog/articles/art_20030321.php
- http://www.openweb.eu.org/articles/html_au_xhtml/

Prologue XML et encodage de caractères

La déclaration XML <?xml> est une composante recommandée du document XHTML. Cette déclaration l'identifie en effet comme appartenant au cadre XML et en décrit la version. Son support inégal dans les principaux navigateurs et ses conséquences sur le rendu CSS incitent parfois à l'omettre. Cependant, comme elle permet de spécifier l'encodage de caractères spéciaux dans le document, choisir de l'ometre expose à un rendu incorrect de ces mêmes caractères. Le cas typique est celui des documents rédigés en français, qui utilisent des caractères spéciaux n'appartenant pas à l'encodage ASCII.

Tout dépend donc de l'encodage choisi :

- Avec les encodages par défaut du XML (*UTF-8* et *UTF-16*), le prologue XML est optionnel, et une balise meta précisera l'encodage approprié pour les anciens navigateurs : <meta http-equiv="content-type" content="text/html; charset=UTF-8">
- Avec d'autres encodages (ISO-8859-1 par exemple), il faut :
    - Soit inclure en tête du document le prologue : <?xml version="1.0" encoding="iso-8859-1"?> ainsi que la balise meta pour les vieux navigateurs qui ignoreront ce prologue.
    - Ou encore spécifier l'encodage au niveau supérieur, c'est à dire dans l'en-tête http serveur.

Utilisation de la déclaration de type de document

Le document doit respecter les normes de validation d'une des trois DTD (Document Type Definition) XHTML ; strict, transitional ou frameset. Une déclaration de type de document doit apparaître dans le document juste avant l'élément <HTML> (qui est l'élément racine de tout document XHTML).

- La DTD stricte n'autorise pas l'utilisation des anciens éléments de présentation (b, i, etc.) mais elle a l'avantage de contraindre le développeur à séparer structure et présentation, avec les facilités de maintenance que cela apporte.
- La DTD transitionnelle est plus permissive et plus proche des anciennes habitudes liées à HTML 3.2. Les balises dépréciées en XHTML strict y sont autorisées : le recours à cette DTD est donc plus facile dans un premier temps, mais avec le défaut de mêler encore partiellement structure et présentation.
- Enfin, la DTD frameset permet l'utilisation des cadres. Ceux-ci tombent en désuétude, mais peuvent se réveler encore utiles dans certains cas très exceptionnels.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Article original publié sur CYBERcodeur.net et OpenWeb le 21 mars 2003 :

- http://www.cybercodeur.net/weblog/articles/art_20030321.php
- http://www.openweb.eu.org/articles/html_au_xhtml/

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

<HTML>, l'élément racine du document XHTML

L'élément racine d'un document doit impérativement être l'élément html et celui-ci se doit d'avoir un espace de nom (namespace) utilisant l'attribut xmlns et une déclaration de la langue utilisée principalement dans le document. Par exemple :

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
```

Gabarits XHTML Strict prêts à l'emploi

Vous pouvez copier/coller ces gabarits directement pour votre usage personnel :

Encodage iso-8859-1 :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
   <title>Votre titre</title>
   <meta http-equiv="Content-Type" content="text/HTML; charset=iso-8859-1" />
</head>
<body>

...votre code...

</body>
</html>
```

Encodage utf-8 ou utf-16 :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
   <title>Votre titre</title>
   <meta http-equiv="Content-Type" content="text/HTML; charset=utf-8" />
</head>
<body>

...votre code...

</body>
</html>
```

Article original publié sur CYBERcodeur.net et OpenWeb le 21 mars 2003 :

- http://www.cybercodeur.net/weblog/articles/art_20030321.php
- http://www.openweb.eu.org/articles/html_au_xhtml/

Gabarits XHTML Transitional prêts à l'emploi

Vous pouvez copier/coller ces gabarits directement pour votre usage personnel :

Encodage iso-8859-1 :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitonnal//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitionnal.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
   <title>Votre titre</title>
   <meta http-equiv="Content-Type" content="text/HTML; charset=iso-8859-1" />
</head>
<body>

...votre code...

</body>
</html>
```

Encodage utf-8 ou utf-16 :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitionnal//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitionnal.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
 <title>Votre titre</title>
 <meta http-equiv="Content-Type" content="text/HTML; charset=utf-8" />
</head>
<body>

...votre code...

</body>
</html>
```

Conclusion

Comme vous pouvez le constater, faire la transition vers le XHTML ne demande pas un grand effort d'adaptation et a le mérite de vous ouvrir toutes grandes les portes de la technologie XML. En y conformant vos documents dès aujourd'hui, vous assurerez la pérennité de vos pages Web et commencerez dès lors à profiter des avantages induits par l'applications des standards Web.

Article original publié sur CYBERcodeur.net et OpenWeb le 21 mars 2003 :

- http://www.cybercodeur.net/weblog/articles/art_20030321.php
- http://www.openweb.eu.org/articles/html_au_xhtml/