Partie I

Le langage XHTML

1

Introduction à XHTML

Il n'y aura pas de HTML 5. C'est ce qu'a confirmé le W3C (World Wide Web Consortium), l'organisme qui édite les recommandations des langages du Web : le HTML est mort en tant que tel. Certes, pendant des années, il a permis à tous de « bricoler » des pages web plus ou moins bien ficelées, mais il était devenu trop permissif, et surtout se caractérisait par un manque de rigueur assez flagrant. La rivalité entre Netscape et Microsoft a entraîné la création de balises propriétaires utilisables uniquement dans l'un des navigateurs, chacun s'ingéniant à créer le gadget qui lui attirerait le plus d'utilisateurs. Cette situation ne faisait que gêner les créateurs de pages web qui étaient obligés de prévoir des solutions alternatives aux balises manquantes en fonction du navigateur client. Il est évident que c'est la pression du e-commerce qui a provoqué la disparition de HTML, et que la naissance de XML (eXtensible Markup Language) l'a précipitée. Le XML aurait pu l'emporter tout de suite (car à terme c'est lui qui restera), mais c'était sans compter avec sa complexité et les problèmes de lecture qu'il pose aux utilisateurs avec des navigateurs d'anciennes générations. Il a donc fallu définir une alternative au tout XML.

Généalogie du XHTML

XHTML (eXtensible HyperText Markup Language) est un langage de balisage (dit aussi langage de marquage) qui permet de structurer le contenu des pages web dans différents éléments. Voilà une définition bien abstraite, reconnaissons-le, mais nous y reviendrons en détail dans la section suivante en présentant la notion de balisage.

Historiquement, les langages de balisage sont issus du langage SGML (Standard Generalized Markup Language) créé en 1986 pour structurer des contenus très divers.

Ce langage s'est révélé trop complexe pour être appliqué tel quel au Web, d'où la nécessité d'en créer une version allégée respectant les mêmes principes essentiels.

L'inventeur du HTML (1992), Tim Berners-Lee, l'avait conçu à l'origine comme un outil de structuration des contenus, principalement textuels, et non pas pour créer des présentations diversifiées. Ce sont les développements successifs, l'essor populaire du Web et la concurrence acharnée entre Netscape et Microsoft pour s'emparer du marché des navigateurs, qui ont détourné HTML de sa vocation première avec l'ajout d'éléments de design qui n'avaient rien à y faire. Voulant faire mieux que l'autre, chacun des deux grands a empilé des couches superflues sur HTML. Il est vrai que l'entrée du Web dans le grand public nécessitait de répondre à une demande d'interfaces graphiques plus esthétiques.

L'absence d'un langage particulier dédié uniquement à la présentation poussait effectivement les webmestres à utiliser tous les moyens pour créer des présentations visuelles agréables. L'apparition de CSS (Cascading Styles Sheets) en 1996 aurait dû résoudre le problème du détournement de HTML de sa destination première. Les mauvaises habitudes étaient prises et la facilité faisait le reste.

L'apparition de HTML 4, et particulièrement de sa version « strict » associée à l'emploi systématique de CSS 2 (publié en 1998), pouvait apporter une solution efficace à ce problème. La création de XML (eXtensible Markup Language) en 1998 et son succès dans de multiples domaines d'application ont conduit le W3C (World Wide Web Consortium) à créer le langage XHTML, non plus comme une nouvelle version de HTML, mais comme une reformulation de HTML en tant qu'application XML. Au niveau des éléments et des attributs disponibles, il existe à vrai dire très peu de différences entre HTML 4 strict et XHTML 1.1.

Les éléments, balises et attributs

Mais au juste comment fonctionne XHTML et qu'est-ce qu'un langage de balisage ?

Vous avez sûrement déjà utilisé un traitement de texte tel que Word. Votre texte peut comprendre des titres, des paragraphes, des images, des tableaux, et vous pouvez utiliser différentes polices de caractères et différentes tailles de caractères dans le même document. Le document final que vous avez réalisé ne laisse apparaître que le résultat de votre mise en page, mais en arrière-plan, votre traitement de texte a enregistré tous les paramètres de mise en page que vous avez utilisés en plus du texte lui-même.

Dans un langage de balisage, tout contenu, qu'il s'agisse de texte, d'image ou d'éléments multimédias les plus divers, doit être renfermé dans un élément. En XHTML, comme dans HTML, chaque élément a un nom déterminé et la liste des éléments utilisables est limitative et clairement définie dans la DTD (Document Type Definition) liée à la version utilisée du langage. C'est la grande différence entre XHTML et XML, langage dans

lequel c'est le programmeur qui crée ses propres éléments selon ses besoins. À quelques exceptions près, un élément a la structure suivante :

```
<nom_element> Contenu </nom_element>
```

Son contenu est précédé par une balise d'ouverture `<nom_element>` et suivi par une balise de fermeture `</nom_element>`. Toutes les balises d'ouverture (ou marqueur) commence par le signe < et se terminent par le signe >. La balise de fermeture fait de même mais le nom de l'élément est précédé d'un slash (/). Les navigateurs interprètent donc les contenus en fonction du nom de l'élément et attribuent un style par défaut à chacun de ses contenus.

Les caractéristiques de chaque élément peuvent être précisées par des informations complémentaires que l'on désigne en tant qu'attributs de l'élément. Il peut s'agir par exemple de la définition de la largeur, de la hauteur ou de l'alignement du contenu. Comme nous le verrons, un certain nombre d'attributs sont communs à quasiment tous les éléments de base dans les sections suivantes.

Les attributs d'un élément sont toujours définis dans la balise d'ouverture et doivent être séparés les uns des autres par au moins une espace typographique. Chaque attribut doit avoir une valeur, contrairement à ce qui se pratiquait dans HTML 4, même s'il ne peut prendre qu'une valeur unique. Aucune valeur n'est donc implicite du moment que l'attribut figure dans la balise d'ouverture. La présence de certains attributs est obligatoire dans quelques éléments particuliers, ce que nous préciserons systématiquement quand ce sera le cas. La plupart du temps, les attributs d'un élément sont facultatifs et c'est au programmeur de déterminer leur définition par rapport au cas qu'il doit traiter. Nombre d'attributs ont une valeur par défaut. Cela signifie que même si on ne les définit pas dans l'élément, celui-ci se comporte comme si nous avions défini explicitement cette valeur. Il est donc important de connaître ce type d'attribut et de ne pas négliger de les définir avec une autre valeur si ce comportement par défaut n'est pas désiré. La valeur de tous les attributs doit être définie entre des guillemets doubles. La syntaxe conforme d'un élément ayant des attributs est donc la suivante :

```
<nom_element attribut1="valeur1" attribut2="valeur2" > Contenu de l'élément
➥ </nom_element>
```

Le contenu d'un élément peut être constitué de texte ou d'autres éléments qui, eux-mêmes, peuvent en contenir d'autres, et ainsi de suite. Cet ensemble d'inclusion constitue la hiérarchie du document XHTML.

Les attributs de base de XHTML

Dans leur quasi-totalité, les éléments disponibles en XHTML ont en commun un ensemble d'attributs ayant chacun le même rôle. Ces attributs se répartissent en trois catégories. Chaque élément peut avoir par ailleurs d'autres attributs particuliers. Quand nous définirons par la suite les différents éléments, nous signalerons s'ils possèdent ces attributs sans rappeler leur définition.

Les attributs courants (noyau)

Ils s'appliquent à quasiment tous les éléments. On compte trois principaux attributs courants et un quatrième encore accepté mais dont l'emploi est déconseillé :

- L'attribut `id`. Il sert à identifier un élément de manière unique en lui donnant un nom, soit pour lui attribuer un style, soit pour y faire référence sans ambiguïté dans un script JavaScript.
- L'attribut `class`. Il contient le nom d'une classe CSS qui contient des définitions de styles. Comme nous le verrons dans la deuxième partie, son usage est très répandu pour affecter des styles ponctuellement à un élément.
- L'attribut `title`. Il contient un texte qui apparaît dans une bulle quand l'utilisateur positionne le curseur quelques instants (ce n'est pas instantané) sur un élément. Le texte qu'il contient peut servir à fournir une information ou une explication sur le rôle de l'élément.
- L'attribut `style`. Il permet de définir un style localement pour un élément donné. Il est encore toléré en XHTML 1.1 mais déconseillé.

Les attributs d'internationalisation

- L'attribut `xml:lang`. Il qui remplace l'attribut `lang` de HTML 4.
- L'attribut `dir`. Il indique le sens de lecture du contenu textuel d'un élément ; il peut prendre les valeurs `ltr` (lecture de gauche à droite) ou `rtl` (de droite à gauche).

Les attributs de gestion d'événements

Ces attributs permettent de gérer les événements dont un élément peut être le siège et qui sont créés par l'utilisateur. Leur contenu est un script écrit en général en langage JavaScript. Les DTD HTML 4 définissent dix attributs de gestion d'événements, y compris pour des éléments qui ne peuvent pas être le siège de ces événements. Il appartient donc aux programmeurs d'effectuer des tests pour vérifier la réalité des événements pour un élément donné. Vous trouverez ci-après la liste des gestionnaires de base et la description de l'événement correspondant.

Tableau 1-1. Les attributs gestionnaires d'événements communs

attribut	Action de l'utilisateur
`onclick`	Clic sur le contenu de l'élément.
`ondblclick`	Double-clic sur le contenu de l'élément.
`onkeydown`	Maintien d'une touche enfoncée.
`onkeypress`	Frappe sur une touche.
`onkeyup`	Relâchement d'une touche enfoncée.

Tableau 1-1. Les attributs gestionnaires d'événements communs *(suite)*

attribut	Action de l'utilisateur
`onmousedown`	Enfoncement d'un bouton de la souris.
`onmousemove`	Le curseur de la souris bouge dans la zone de l'élément.
`onmouseout`	Le curseur de la souris quitte la zone de l'élément.
`onmouseover`	Le curseur de la souris est au-dessus de la zone de l'élément.
`onmouseup`	Relâchement d'un bouton de la souris au-dessus de la zone de l'élément.

D'autres attributs gestionnaires d'événements sont spécifiques à certains éléments, que nous citerons au fur et à mesure de notre étude de ceux-ci.

Intérêt des spécifications

La création du XHTML ne consiste pas seulement en la redéfinition de quelques règles syntaxiques pour aménager le langage HTML. S'il ne s'agissait que de cela, cet ouvrage n'aurait pas lieu d'être. Il s'agit bien plus selon moi d'un changement de pensée et d'organisation qui doit s'opérer dans la création des pages web.

Une page web créée avec XHTML doit être pensée en distinguant deux parties :

- Un contenu, structuré au moyen des éléments XHTML (grandes divisions, titres, paragraphes, tableaux, images et liens, etc.). À ce stade, et même s'il en a déjà une idée, le créateur ne doit pas nécessairement avoir une idée définitive de la présentation finale. Il lui faut maîtriser principalement l'organisation des informations à fournir à un utilisateur.
- Une feuille de style CSS, définissant la mise en page de ces éléments en fonction du média qui va opérer le rendu du contenu (polices et tailles de caractères, bordures, marges, couleurs, positionnement dans la page, etc.). Les médias se diversifient en effet de plus en plus en devenant des éléments portables dotés de petits écrans, et il n'est pas impossible que le traditionnel écran d'ordinateur ne soit plus à l'avenir le principal vecteur d'affichage d'une page web. La séparation du contenu et de la présentation étant réalisée, il est possible d'associer à chaque média une feuille de style adaptée au terminal.

L'utilisation de ces méthodes présente les avantages suivants :

- Une meilleure organisation du contenu.
- Une meilleure qualité du code et plus grande rapidité d'affichage sur les navigateurs récents (Firefox, Mozilla, Internet Explorer, Netscape…).
- Une réduction des coûts de développement et de maintenance des sites web ainsi qu'une réutilisabilité accrue et rapide du code. En effet, en ayant respecté les principes

précédents, il est très facile de modifier rapidement toute la présentation d'une page sans toucher au code XHTML.

Les standards XHTML et CSS sont aujourd'hui incontournables pour tous ceux qui veulent concevoir un site web de manière professionnelle, et tous les étudiants en informatique et les professionnels du Web se doivent d'acquérir ou de mettre à jour leurs connaissances sur ces techniques.

Règles de base XHTML

Un document bien formé

Un document XHTML doit respecter certaines règles simples :

- Les éléments et les attributs sont sensibles à la casse et doivent être écrits en minuscules. Par exemple, `<body>` et non plus `<BODY>` comme en HTML.
- Les éléments non vides doivent avoir obligatoirement une balise d'ouverture et une balise de fermeture. Par exemple, on ne doit plus écrire :

```
<ol>
  <li>Item 1
  <li>Item 2
```

mais le code suivant :

```
<ol>
  <li>Item1 </li>
  <li>Item2 </li>
</ol>
```

- Les éléments vides ne comportent qu'une seule balise et doivent se terminer par les caractères `/>` précédés d'une espace pour marquer la fin de l'élément. Par exemple, il ne faut plus écrire :

```
<img src= "monimage.gif"> <hr> <br>
```

mais le code suivant :

```
<img src= "monimage.gif" /> <hr /> <br />
```

- Les éléments ne doivent pas se chevaucher. Ils doivent donc obéir au principe premier-ouvert-dernier-fermé. Dans ce cas, le premier élément est le parent du second et celui-ci est enfant du premier. Par exemple, le code suivant est incorrect :

```
<div> Cette division contient un titre <h1> Important ! </div> </h1>
```

et doit être remplacé par :

```
<div> Cette division contient un titre <h1> Important ! </h1></div>
```

- Tous les attributs doivent avoir une valeur incluse entre des guillemets doubles (`"`). Les différents attributs du même élément doivent être séparés par au moins une espace. Par exemple, il ne faut plus écrire :

```
<p class=styleperso title=attention> Texte important</p>
```

  mais le code suivant :

```
<p class="styleperso" title="attention" > Texte important</p>
```

- À tous les attributs utilisés doit être donnée une valeur, y compris ceux dont la valeur est unique. Par exemple, il ne faut plus écrire :

```
<input type= "checkbox" checked disabled />
```

  mais le code suivant :

```
<input type= "checkbox" checked="checked" disabled="disabled" />
```

- L'attribut `name` qui servaient à identifier certains éléments (`<a>`, `<form>`, par exemple) est supprimé et doit être remplacé par l'attribut `id`.

- Les scripts et les feuilles de style qui contiennent les caractères `<` et `&` peuvent figurer dans des sections CDATA de la façon suivante :

```
<script type="text/javascript">
  < ![CDATA[
    Code du script…
  ]]>
</script>
```

  Ces sections n'étant pas actuellement reconnues par les navigateurs, nous ne les utiliserons pas dans cet ouvrage, mais il faudra en tenir compte dans un avenir proche. Une autre solution efficace consiste à inclure les scripts ou les feuilles de style contenant ces caractères dans des fichiers séparés et à les inclure à l'aide de l'élément `<link>` sur lequel nous reviendrons en détail par la suite.

Un document conforme

Un document XHTML se doit également de respecter les règles d'inclusion des éléments les uns dans les autres, telles qu'elles sont définies dans la DTD choisie. En effet, une DTD définit la liste limitative de tous les éléments XHTML utilisables et énumère ceux qui peuvent y être inclus. Le respect de ces contraintes est impératif pour que le document soit déclaré conforme à la DTD. Vous trouverez à cet effet tout au long de cet ouvrage lors de la description des éléments, et dans l'annexe A, non pas le texte des DTD XHTML 1 qui est particulièrement difficile à lire, mais son interprétation, pour chaque élément, sous la forme de la liste de ses éléments enfants (ceux qu'il peut inclure) et de ses éléments parents (ceux dans lesquels il peut être inclus).