Résumé. L'objectif de ce poly est, d'une part, de se familiariser avec la notion d'emboîtement (tant logique que graphique) des éléments en XHTML et, d'autre part, de parcourir les éléments de listes et de tableaux, qui permettent de structurer une collection d'éléments.

1 Organisation d'un document

1.1 Description logique d'un document

Comment décrire un document dans un certain formalisme, de sorte à ce que n'importe quel éditeur (indépendamment du support d'édition) puisse le retranscrire fidèlement à partir de sa description ? Si l'on prend l'exemple de ce document, il contient un titre principal, un ensemble de paragraphes, des bouts de phrases mis en exergue, un ensemble de sous-titres, un ensemble de listes énumérées, de figures, de tableaux, une annexe, ... Ces contenus sont ordonnés et hiérarchisés, en sections (titres 1, 2, 3, ...), sous-sections (titres 1.1, 1.2, 1.3, ...), sous-sous-sections, paragraphes, ... Pour décrire ce document, il faut donc, d'une part, identifier ses différentes *composantes* et, d'autre part, retranscrire leurs *dépendances*. Par exemple : le titre 1.2, le paragraphe et la figure qui le suivent, forment la section 1.2, et se placent *après* le titre 1.1, le paragraphe et la figure qui constituent la section 1.1. L'ensemble de ces deux sections est *contenu dans* la section 1, *etc.* Ainsi, le document peut être décrit par l'intermédiaire d'une *arborescence*, dont la racine représente le document lui-même, dont les feuilles sont les contenus élémentaires (*ex.* : un titre), et les nœuds représentent un ensemble de contenus. Les fils d'un nœud sont des éléments ou des regroupements d'éléments contenus dans ce nœud. L'éventuel deuxième fils d'un nœud est *le successeur* de son premier fils, et a lui-même comme successeur l'éventuel troisième fils de son père. Une représentation logique (partielle) de ce document est présentée dans la figure 1.

document	⟵	{ *propriétés*, *contenu* }
propriétés	⟵	{ *titre*, *auteur*, *date*, ... }
contenu	⟵	{ *résumé*, *section 1*, *section 2*, *section 3*, *section 4*, *annexe* }
résumé	⟵	{ *paragraphe de résumé* }
section 1	⟵	{ *titre 1*, *sous-section 1.1*, *sous-section 1.2*}
sous-section 1.1	⟵	{ *titre 1.1*, *paragraphe 1.1*, *figure 1*}
sous-section 1.2	⟵	{ *titre 1.2*, *paragraphe 1.2*, *figure 2*}
...		

Figure 1: Structuration logique d'un document

1.2 Traduction (X)HTML

XHTML permet de retranscrire la structure arborescente d'un document : tout document XHTML a un élément *racine* "html" (qui représente le document) ; cet élément a *deux fils*, qui sont un élément "head" et un élément "body" ; par la suite, tout élément de contenu est, directement ou indirectement, contenu dans l'élément "body". Les éléments qui y sont *directement* contenus sont appelés *fils* ou *descendants directs* ; ceux qui y sont contenus *indirectement* (par exemple, un paragraphe contenu dans une division elle-même contenue dans "body") sont appelés *descendants* indirects. L'ensemble de ces éléments forme *les descendants* de l'élément " body". La notion de descendance, fondée sur la relation de *contenant à contenu*, est définie pour tous les éléments. Ainsi, par exemple, l'élément "title" est un fils de "head", et un descendant de "html". De même, tout élément de contenu est un

descendant de “body”. On peut alors dessiner *l'arborescence* des éléments d'un document XHTML : la racine de l'arbre est l'élément “html”, puis les fils d'un élément de l'arbre est l'ensemble des éléments qui sont ouverts (directement) dans l'élément considéré. Si l'on reprend l'exemple de ce document, une représentation XHTML pourrait en être la représentation proposée en figure 2.

xhtml	⟵	{ *head*, *body* }
head	⟵	{ *title*, ... }
body	⟵	{ *div id="resume"*, *div id="sec-1"*, *div id="sec-2"*, *div id="sec-3"*, *div id="sec-4*, *div id="annexe"* }
div id="resume"	⟵	{ *p id="resume"* }
div id="sec-1"	⟵	{ *h1 id="titre-1"*, *div id="sec-1-1"*, *div id="sec-1-2"*, *div id="sec-1-3"* }
div id="sec-1-1"	⟵	{ *h2 id="titre-1-1"*, *div id="sec-1-1-1"*, *div id="sec-1-1-2"* }
div id="sec-1-1-1"	⟵	{ *h3 id="titre-1-1-1"*, *p*, *table id="tab-1"* }
...		

Figure 2: Structuration logique en XHTML

2 De la structuration logique à la représentation graphique

La structuration logique du document n'est pas sans lien avec sa représentation graphique :

1. de la dépendance *logique* entre éléments, on déduit *l'héritage* des propriétés ;
2. chaque élément étant graphiquement représenté par une *boîte*, l'emboîtement des éléments *logiques* induit celui de leur *représentation graphique*.

2.1 Modèle de boîte

En XHTML, chaque élément *logique* de contenu définit, lors de sa publication, un composant *graphique* (qu'il s'agisse d'un élément “p”, “h3”, “img”, “span”,...). Ce composant n'est autre qu'une boîte (un rectangle) qui prend la taille nécessaire à l'affichage de son contenu. Par exemple,

l'élément : attention !
définit graphiquement la boîte : **attention !**

Tandis que le paragraphe :
Mais attention ! à bien structurer d'abord son document
définit graphiquement la boîte :

Mais **attention !** à bien structurer d'abord son document

Si l'on dessine les boîtes définies par les deux éléments, on se rend compte qu'elles sont emboîtées :

Mais **attention !** à bien structurer d'abord son document

La boîte d'un élément définit un espace graphique dont on peut préciser certaines propriétés, notamment : ses dimensions (largeur, hauteur), ses espacements (vis-à-vis de son contenant, vis-à-vis de son contenu), sa bordure et, bien sûr, son positionnement (dans son contenant). Une difficulté à laquelle est confronté le développeur XHTML est que ces propriétés ne sont pas interpétées de la même façon par tous les navigateurs... mais nous verrons tout cela plus tard. Pour l'instant, on se contente de placer les éléments les uns à la suite des autres dans le flux XHTML, et l'on ne gère pas explicitement le dimensionnement des boîtes.

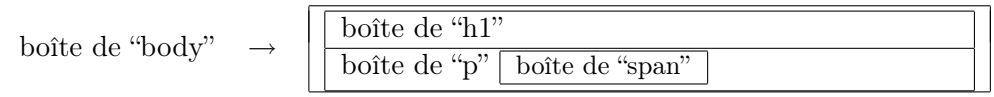
On peut donc voir une page web comme un enchevêtrement de boîtes graphiques qui découle de l'organisation logique du document. Ces boîtes s'agencent *à l'intérieur* les unes des autres lorsqu'elles correspondent à des éléments *fils et père*, *à la suite* les unes des autres lorsqu'elles correspondent à

des élément *frères*. Il existe différents types de boîte, qui vont (notamment) définir la façon dont les boîtes de deux éléments *frères* vont se placer *à la suite* l'une de l'autre : l'une *en-dessous* de l'autre, ou l'une *à côté* de l'autre.

2.1.1 Emboîtement et flux XHTML

Pour commencer, on distinguera deux types de boîte : le type *bloc*, et le type *en ligne*. Une boîte de type *bloc* se place *en-dessous* de la boîte définie par l'élément précédent (frère aîné) et force la boîte de l'élément suivant (frère cadet) à se placer en-dessous : elle occupe ainsi toute la largeur disponible et opère un passage à la ligne *avant* et *après* son affichage. Une boîte de type *en ligne* se placera elle *à la suite* de la boîte définie par l'élément précédent : elle occupe alors la largeur juste nécessaire à l'affichage de son contenu, sans opèrer de passage à la ligne, ni avant, ni après son affichage. Par défaut, les *éléments de niveau bloc* définissent des boîtes de type *bloc* et les *éléments de niveau texte* définissent des boîtes de type *en ligne*. La largeur disponible pour l'affichage d'une boîte de type bloc est la largeur du contenu de la boîte qui la contient (élément père). Sauf spécification contraire, la hauteur d'une boîte, quel qu'en soit le type, s'ajuste à son contenu (*ie.*, une boîte occupe toujours la hauteur nécessaire à l'affichage de son contenu).

Donnons un petit exemple, dans lequel on suppose que tous les éléments définissent des espacements et des bordures nulles, c'est-à-dire que les dimensions de leur boîte et de leur contenu coïncident. L'élément "body" définit une boîte qui prend toute la largeur de la page et la hauteur de son contenu. Deux éléments "h1" et "div" fils de "body" définissent chacun une boîte de type *bloc* qui prend la largeur de la boîte de "body". Un élément "p" fils de cet élément "div" définit une boîte de type *bloc* qui prend la largeur de la boîte de "div". Si ce paragraphe a un élément "span" pour fils, cet élément "span" définit une boîte de type *en ligne* qui occupe la largeur juste nécessaire à l'affichage de son contenu. Ainsi, les boîtes de "h1" et de "div" sont contenues dans la boîte de "body" (et la boîte de "div" se place en-dessous de celle de "h1"), la boîte de "p" est contenue dans celle de "div", la boîte de "span" est contenue dans celle de "p" :



2.1.2 Définition du type de boîte d'un élément

La propriété "display" permet de choisir le type de boîte pour l'affichage d'un élément. Supposons que l'on dispose de deux éléments successifs $fils_1$ et $fils_2$ (par exemple, deux éléments "img", ou deux éléments "li") d'un même élément parent (par exemple, d'un élément "div", ou d'un élément "ul"). On peut vouloir que les deux fils soient affichés l'un à côté de l'autre (comportement par défaut pour "img" qui est de niveau texte) ou, au contraire, l'un en-dessous de l'autre (comportement par défaut pour "li" qui est de niveau bloc). Pour que les deux fils s'affichent comme on le souhaite, il suffit de donner à leur propriété "display" la bonne valeur : pour les afficher *côte à côte*, il faut les insérer avec la valeur *inline* (de sorte à ce qu'ils définissent des boîtes de type *en line*), tandis que pour les afficher *l'un en-dessous de l'autre*, il faut les insérer avec la valeur *block* (de sorte à ce qu'ils définissent des boîtes de type *bloc*). Parmi les valeurs que peut prendre la propriété "display", mentionnons pour l'instant :

1. *none* : le contenu de l'élément n'est pas affiché et aucun emplacement n'est pris par le contenu (l'élément définit donc une boîte de surface nulle) ;

2. *block* : l'élément définit une boîte de type *bloc* (et se place donc *en-dessous* de l'élément précédent) ;

3. *inline* : l'élément définit une boîte de type *en ligne* (et se place donc, comme du texte, *à côté* de l'élément précédent).

```
<body>
   <h1>CV de Toto</h1>
   <h2 id="titre-sec-1">Objectifs</h2>
   <div id="contenu-sec-1">
      <p id="p-obj">
         &Eacute;voluer <span>&agrave; moyen terme</span> vers un poste de consultant
         en syst&egrave;mes d'information...
      </p>
   </div>
   <h2 id="titre-sec-2">Parcours professionnel</h2>
   <div id="contenu-sec-2">
      <h3 id="h3-0002">2000-2002 - Soci&eacute;t&eacute; TATA</h3>
      <p id="p-0002">Ing&eacute;nieur &eacute;tudes &amp; d&eacute;veloppement...</p>
      <h3 id="h3-0204">2002-2004 - Soci&eacute;t&eacute; TITI</h3>
      <p id="p-0002-cs">Concepteur de sites web commerciaux...</p>
      <p id="p-0002-cp">Chef de projets...</p>
      <h3 id="h3-04--">depuis mars 2004 - Soci&eacute;t&eacute; TUTU</h3>
      <p id="p-04---dp">Directeur de projets architectures n-tiers...</p>
   </div>
   <h2 id="titre-sec-3">Formation</h2>
   <div id="contenu-sec-3">
      ...
   </div>
   ...
</body>
```

Figure 3: Illustration

2.2 Héritage des propriétés

Les propriétés de mise en forme sont définies tout au long d'un document XHTML, dans tout élément. Ces propriétés ont donc des valeurs par défaut, qui peuvent être spécifiques aux éléments (*ex.* : la couleur de police par défaut est *black* pour tous les éléments ; l'épaisseur de trait des caractères est *normal* pour la majorité des éléments, mais *bold* pour "strong", "h1", "th" ; la propriété "display" vaut *block*, *inline*, *table* pour les éléments "div" (de niveau bloc), "span" (de niveau texte), "table", respectivement). Par ailleurs, on peut spécifier à tout moment (c'est-à-dire, dans n'importe quel élément) la valeur d'une propriété. Comment connaître alors, à un moment donné, la valeur d'une propriété ? Tout d'abord, cela dépend du type de propriété, essentiellement : si la propriété s'applique *à l'élément* lui-même (*ex.* : "id", "display"), ou à *son contenu* (*ex.* : "bakground-color", "text-align"). Si la propriété s'applique au contenu, alors la valeur de la propriété de l'élément parent est *héritée* par ses éléments fils, dont les propres fils vont de nouveau hériter, et ainsi de suite. Ainsi, l'arborescence des éléments joue un rôle capital dans l'apparence de vos pages !

3 Illustration

Dans cette section, on comente essentiellement le code XHTML proposé dans la figure 3.

3.1 Structuration logique du document

Quelques liens de parenté sur ce code :

1. Les éléments “h1”, “h2 - titre-sec-1”, “div - contenu-sec-1”, “h2 - titre-sec-2” “div - contenu-sec-2”, “h2 - titre-sec-3” “div - contenu-sec-3” sont les fils successifs de l’élément “body”.
2. L’élément “p - p-obj” a l’élément “div - contenu-sec-1” pour père et un élément “span” pour fils.
3. L’élément “div - contenu-sec-2” a 7 fils, répartis en trois éléments “h3” et 4 éléments “p”.
4. Les éléments “div - contenu-sec-3” et “div - contenu-sec-1” sont des éléments de même niveau.
5. L’élément “span” est un descendant de l’élément “div - contenu-sec-1”.
6. L’élément “body” est un ascendant de tous les autres éléments.

L’arborescence complète (partant de “body”) est donnée dans la figure 4.

racine	*niveau 1*	*niveau 2*	*niveau 3*
body ⟵	h1		
	h2 - titre-sec-1		
	div - contenu-sec-1 ⟵	p - p-obj ⟵	span
	h2 - titre-sec-2		
	div - contenu-sec-2 ⟵	h3 - h3-0002	
		p - p-0002	
		h3 - h3-0204	
		p - p-0204-cs	
		p - p-0204-cp	
		h3 - h3-04--	
		p - p-04---dp	
	h2 - titre-sec-3		
	div - contenu-sec-3		

Figure 4: Arborescence des éléments

3.2 Emboîtements

Dans l’annexe de ce document, le contour des éléments est tracé, de sorte à rendre visible l’emboîtement des blocs définis par les éléments. Le code de la figure 3 est reproduit trois fois :

1. une première fois, sans modifier la valeur par défaut de la propriété “display” ;
2. une deuxième fois, en mettant systématiquement “display” à la valeur *inline* ;
3. une troisième fois, en mettant “display” à la valeur *block* pour l’élément de niveau texte “span”.

Ces jeux sur la valeur de la propriété “display” sont discutés dans la section suivante.

3.3 Jeux d’affichage

1. Si l’on publie le code XHTML tel quel, à l’exception de l’élement “span” qui définit une boîte de type *en ligne*, tous les éléments définissent une boîte de type *bloc* (les éléments de niveau bloc - *resp.*, texte - s’affichent par défaut avec la valeur *block resp.*, *inline* - de la propriété “display”). Cela produit l’affichage :

CV de Toto
Objectifs
Évoluer à moyen terme vers un poste de consultant en systèmes d’information [...]

Et les emboîtements :

CV de Toto

Objectifs

Évoluer | à moyen terme | vers un poste de consultant en systèmes d'information [...]

2. Si l'on ajoute pour chaque élément l'attribution :

   style="display:inline;"

   alors les contenus de tous les éléments s'affichent les uns à la suite des autres, comme s'il s'agissait d'un seul et même paragraphe :

   CV de Toto **Objectifs** Évoluer à moyen terme vers un poste de consultant en systèmes d'information [...]

   Et les boîtes définies deviennent :

   CV de Toto | **Objectifs** | Évoluer | à moyen terme | vers un poste [...]

3. Si au contraire l'on ajoute dans l'élément "span" du premier paragraphe l'atribution :

   style="display:block;"

   alors le navigateur opère un passage à la ligne avant et après l'élément :

   # CV de Toto
   ## Objectifs
   Évoluer
   à moyen terme
   vers un poste de consultant en systèmes d'information [...]

3.4 Priorités des valeurs de propriétés

Dans les figures 5 et 6, on reprend une partie du code de la figure 3 en jouant cette fois-ci sur la valeur de la propriété "bakground-color", qui s'applique *au contenu* d'un élément. On rappelle que la valeur d'une propriété est *héritée de père* (ou contenant) *en fils* (ou contenu). Alors, si l'on regarde le code proposé dans la figure 5 :

La couleur de fond pour "body" est *rouge* (affectation via l'attribut "style").
⇒ La couleur de fond pour ses fils est également *rouge* :
"h1 - h1-cv", "h2 - h2-obj", "div - div-obj", "h2 - h2-pp", "div - div-pp"
⇒ La couleur de fond pour le fils "p - par-obj" de l'élément "div - div-obj" est *rouge*.
La couleur de fond pour les fils de l'élément "div - div-pp" est *rouge* :
"h3 - h3-00-02" et "p - par-00-02"

Bien sûr, si un élément fils spécifie lui-même la valeur d'une propriété, c'est cette valeur qui sera prise en compte (voir la figure 6).

4 Éléments liste et tableau

4.1 Listes

Les listes permettent de définir un bloc (la liste elle-même), lui-même subdivisé en sous-blocs (les items de la liste). Évidemment, la vocation première d'une liste XHTML est de retranscrire une liste, éventuellement ordonnée, de données (structuration du contenu même). Mais les listes constituent également un outil fort pratique pour travailler sur la structuration de la navigation et du rendu

```
<body style="background-color:#FF0000;">
    <h1 id="h1-cv">CV de Toto</h1>
    <h2 id="h2-obj">Objectifs</h2>
    <div id="div-obj">
        <p id="par-obj">&Eacute;voluer <span>&agrave; moyen terme</span> [...]</p>
    </div>
    <h2 id="h2-pp">Parcours professionnel</h2>
    <div id="div-pp">
        <h3 id="h3-00-02">2000-2002 - Soci&eacute;t&eacute; TATA</h3>
        <p id="par-00-02">Ing&eacute;nieur &eacute;tudes &amp; d&eacute;veloppement...</p>
    </div>
</body>
```

Figure 5: Héritage des valeurs de propriétés - 1

élément	**background-color**
`<body style="background-color:#FF0000;">`	rouge
`<style="background-color:#00FF00;"h1 id="h1-cv">CV de Toto</h1>`	vert
`<h2 id="h2-obj">Objectifs</h2>`	rouge par “boby”
`<div style="background-color:#0000FF;" id="div-obj">`	bleu
`<p id="par-obj">&Eacute;voluer <span>&agrave; moyen terme</span> [...]</p>`	bleu par “div”
`</div>`	
`<h2 id="h2-pp">Parcours professionnel</h2>`	rouge par “body”
`<div id="div-pp">`	rouge par “body”
`<h3 style="background-color:#0000FF;" id="h3-00-02">`	bleu
`2000-2002 - Soci&eacute;t&eacute; TATA`	
`</h3>`	
`<p id="par-00-02">Ing&eacute;nieur`	rouge par “body”
`<span style="background-color:#00FF00;">`	vert
`&eacute;tudes & d&eacute;veloppement...</span>`	
`</span>`	
`</p>`	
`</div>`	
`</body>`	

Figure 6: Héritage des valeurs de propriétés - 2

visuel (*ex.* : création de menus pour la navigation et plus généralement, organisation d'un ensemble d'éléments dans l'espace). Un avantage des listes est que l'on n'a pas besoin d'en dénifir la taille *a priori* : la taille de la liste est donnée précisément par le nombre d'items qui y sont insérés.

4.1.1 Types de listes

Il existe trois familles de listes :

1. Liste *non ordonnée* ou *liste à puce* : élément “ul”. Ces listes permettent de représenter une collection d'entités (avec puce et indentation).

2. Liste *ordonnée* : élément “ol”. Ces listes permettent de représenter une succession d'entités numérotées (avec numérotation donc, et indentation).

3. Liste *de définition* : élément “dl”. Ces listes permettent de représenter un dictionnaire, c'est-à-dire, une collection de termes ou d'intitulés suivis de leur définition ou contenu. Les items sont toujours indentés, mais ces listes ne comportent ni puce, ni numéro.

4.1.2 Syntaxe

Les items de liste des éléments “ol” et “ul” sont décrits par l'élément “li” ; un élément “ol” ou “ul” doit contenir au moins un élément “li”. Les items des listes “dl” sont des couples d'éléments “dt” (terme ou intitulé) et “dd” (description ou contenu) ; un élément “dl” doit contenir au moins un couple d'éléments (“dt”, “dd”). Ainsi, les trois catégories de liste suivent la syntaxe suivante :

liste à puce	**liste ordonnée**	**liste de définition**
  - item_1 - item_2 ... - item_ℓ 	 - item_1 - item_2 ... - item_ℓ 	<dl> <dt>item_1</dt> <dd>item_1</dd> <dt>item_2</dt> <dd>item_2</dd> ... <dt>item_ℓ</dt> <dd>item_ℓ</dd> </dl>
($\ell \geq 1$)	($\ell \geq 1$)	($\ell \geq 1$)

On peut imbriquer des listes de toute forme. Par exemple, pour un sommaire, on peut vouloir utiliser une numérotation hiérarchisée (emboîtement d'éléments “ol”). On peut également vouloir introduire une liste à puce à l'intérieur d'une définition (emboîtement d'éléments “dl” et “ul”) ; par exemple, pour associer différentes définitions à un même terme, en fonction de son contexte d'utilisation ; ou encore, dans le cadre d'un index, pour lister, pour un même terme, les différentes pages qui le réfèrent.

4.1.3 Propriétés de style

Propriété “display” Les éléments “li” d'une même liste “ul” ou “ol”, les éléments “dt” et “dd” d'une même liste “dl”, sont des éléments de même niveau puisqu'ils partagent le même parent qu'est l'élément liste. Si l'on souhaite afficher les items à la suite *verticalement* les uns des autres, il faut insérer les items de liste avec les valeurs *block* ou *list-item* de la propriété “display”. Si, en revanche, on souhaite afficher les items sur la même ligne (*ie.*, à la suite *horizontalement* les uns des autres), alors il faut les insérer avec la valeur *inline* de la propriété “display”. La valeur par défaut de la propriété “display” est *list-item*, qui seule permet l'affichage d'une puce ou numérotation. C'est d'ailleurs là la seule différence entre les valeurs *block* et *list-item* : les boîtes créées sont les mêmes dans les deux cas, sauf que la valeur *block* ne tient pas compte de la puce (ou numérotation).

Propriété “list-style-type” (puce et numérotation) pour “ul”, “ol”, “li”

1. Valeur de “list-style-type” pour le type de puce (listes “ul”) :

valeur	*none*	*square*	*disc*	*circle*	*inherit*
puce	aucune	carré	disque	cercle	héritée (*)

2. Valeur de “list-style-type” pour le type de numérotation (listes “ol”) :

valeur	*none*	*upper-alpha*	*lower-alpha*	*upper-roman*	*upper-roman*	*decimal*	*inherit*
num.	aucune	A,B,C,...	a,b,c,...	I,II,III,...	i,ii,iii,...	1,2,3,...	héritée(*)

(*) la valeur de la propriété est héritée de l'élément parent.

Propriété “list-style-position” (alignement du texte) pour tous les éléments Les listes proposent deux façons d'aligner à gauche le texte des items de liste (en cas de retour à la ligne) :

1. alignement *après* la puce (le texte de la deuxième ligne est donc aligné au même niveau que celui de la première ligtne) : c'est la valeur *outside* ;

2. alignement *au niveau de la* la puce (le texte de la deuxième ligne est donc aligné plus à gauche que ne l'est le texte de la première ligtne) : c'est la valeur *inside*.

Les valeurs possibles pour la propriété "list-style-position" sont donc *outside* (valeur par défaut), *inside* et *inherit* (héritage de la valeur de propriété par l'élément parent).

4.2 tableaux

Les tableaux XHTML offrent une forte potentialité de structuration des contenus web : si leur première vocation est sans doute de permettre la représentation de tableaux de données, ils permettent également d'organiser matriciellement des blocs de contenu. Notons que les tableaux sont des éléments de niveau bloc particuliers (la valeur par défaut de la propriété "display" pour un tableau est la valeur "table"), puisqu'un tableau définit une boîte (avec passages à la ligne avant et après) dont la largeur est définie par les attributs et/ou le contenu du tableau.

4.2.1 Structuration du contenu d'un tableau

Des lignes et des cellules

table	**tr**	**td**
<table>	<table>	<table>
...	<tr>	<tr>
</table>	...	<td>...</td>
	</tr>	<td>...</td>
	<tr>	</tr>
	...	<tr>
	</tr>	<td>...</td>
	</table>	</tr>
		</table>

Un tableau est un ensemble de lignes et de colonnes, dont les intersections défissent les cases du tableau. Pour sa représentation en xhtml, un tableau est défini par son élément racine "table" (balises ouvrante et fermante), par les éléments de ligne "tr" qu'il contient (balises ouvrante et fermante) et enfin, par les éléments de cellule "td" qui sont contenus par ses lignes (balises ouvrante et fermante). En d'autres termes, un tableau en XHTML est vu comme une collection de lignes, chaque ligne étant elle-même vue comme une collection de cellules.

Le contenu d'un tableau est donc décrit par ses lignes, et la structuration en colonnes est implicitement définie par les cellules du tableau. Néanmoins, il existe un élément de description des colonnes d'un tableau : c'est l'élément "col". Cet élément a pour seule vocation d'attribuer un certain comportement commun à une ou plusieurs colonnes (largeur, propriétés de style), mais ne permet, ni d'introduire du contenu, ni de structurer celui-ci. Cet élément est décrit plus tard, en section 4.2.1.

Un second élément de cellule, "th", définit des *cellules d'entête* ; cela permet de donner un formatage particulier aux cellules qui sont créées avec cet élément (comportement par défaut : affichage du contenu en caractères gras et alignement centré dans les cellules). Un exemple de tableau à 4 lignes et 3 colonnes est donné dans la figure 7, dont le rendu visuel sera le suivant :

caractère	**code HTML**	**code ASCII**
&	&amp;	&#38;
€	&euro;	&#128;
Ô	&Ocirc;	&#212;

Un tableau peut enfin contenir un élément "caption", qui permet de décrire la légende du tableau : <caption> *légende du tableau* </caption>. Attention : si l'on spécifie la légende, alors l'élément "caption" doit être ouvert *immédiatement* après l'ouverture du tableau. Cela induirait sur l'exemple de la figure 7 le code suivant :

```
<table>
  <caption>Caract&egrave;res sp&eacute;ciaux en XHTML</caption>
  <tr>
    ...
</table>
```

```
<table>
  <tr>                                  <!-- première ligne -->
    <th>caract&egrave;</th>             <!-- première colonne ⇒ première cellule -->
    <th>code HTML</th>                  <!-- deuxième colonne ⇒ deuxième cellule -->
    <th>code ASCII</th>                 <!-- troisième colonne ⇒ troisième cellule -->
  </tr>                                 <!-- fin de la première ligne -->
  <tr>                                  <!-- deuxième ligne -->
    <th>&amp;amp;</th>                  <!-- première colonne ⇒ quatrième cellule -->
    <td>&amp;amp;amp;</td>              <!-- deuxième colonne ⇒ cinquième cellule -->
    <td>&amp;amp;&amp;#35;38;</td>       <!-- troisième colonne ⇒ sixième cellule -->
  </tr>                                 <!-- fin de la deuxième ligne -->
  <tr>                                  <!-- troisième ligne -->
    <th>&amp;#128;</th>                 <!-- première colonne ⇒ septième cellule -->
    <td>&amp;amp;euro;</td>             <!-- deuxième colonne ⇒ huitième cellule -->
    <td>&amp;amp;&amp;#35;128;</td>      <!-- troisième colonne ⇒ neuvième cellule -->
  </tr>                                 <!-- fin de la troisième ligne -->
  <tr>                                  <!-- quatrième ligne -->
    <th>&amp;Ocirc;</th>                <!-- première colonne ⇒ dixième cellule -->
    <td>&amp;amp;Ocirc;</td>            <!-- deuxième colonne ⇒ onzième cellule -->
    <td>&amp;amp;&amp;#35;212;</td>      <!-- troisième colonne ⇒ douzième cellule -->
  </tr>                                 <!-- fin de la quatrième ligne -->
</table>                                <!-- fermeture du tableau -->
```

Figure 7: Tableau

Regroupement de cellules Une cellule (élément “td” ou “th”) s’étend par défaut sur une ligne et une colonne, mais on pourrait vouloir spécifier qu’une cellule s’étend sur plusieurs lignes et/ou sur plusieurs colonnes. Ce sont les attributs “rowspan” et “colspan” de l’élément “td” (ou “tr”) qui permettent de spécifier respectivement le nombre de lignes et le nombre de colonnes qui sont occupées par une cellule. Sur l’exemple précédent, on voudrait ajouter une ligne pour préciser que certains caractères spéciaux ne disposent pas de code HTML ; plus précisément, on voudrait obtenir le tableau suivant :

caractère	**code HTML**	**code ASCII**
&	&amp;	&#38;
#	(*)	&#35;
€	&euro;	&#128;
Ô	&Ocirc;	&#212;
(*) pas de code html pour ces caractères spéciaux		

Pour ce faire, le texte de la dernière ligne doit s’étendre sur toutes les colonnes du tableau. Sinon, on obtiendrait :

caractère	**code HTML**	**code ASCII**
&	&amp;	&#38;
#	(*)	&#35;
€	&euro;	&#128;
Ô	&Ocirc;	&#212;
(*) pas de code html pour ces caractères spéciaux		

Il faut donc spécifier au moment de la description de la cellule qui contient ce texte la valeur de l’attribut “colspan” :

```
<tr><td colspan="3">(*) pas de code html pour ces caract&egrave;res sp&eacute;ciaux</td></tr>
```

Le code complet est donné en figure 8, où les valeurs des attributs “colspan” et “rowspan” sont spécifiés pour chaque cellule.

Regroupement de lignes Pour décrire plus finement encore un tableau, il existe des éléments permettant de déclarer des groupes de lignes : il s’agit des élément “thead” (permettant de regrouper des lignes

```
<table>
  <tr>
    <th rowspan="1" colspan="1">caract&egrave;re</th>
    <th rowspan="1" colspan="1">code HTML</th>
    <th rowspan="1" colspan="1">Code ASCII</th>
  </tr>
  <tr>
    <th rowspan="1" colspan="1">&amp;</th>
    <td rowspan="1" colspan="1">&amp;amp;</td>
    <td rowspan="1" colspan="1">&amp;&#35;38;</td>
  </tr>
  <tr>
    <th rowspan="1" colspan="1">&euro;</th>
    <td rowspan="1" colspan="1">&amp;euro;</td>
    <td rowspan="1" colspan="1">&amp;&#35;128;</td>
  </tr>
    <tr>
    <th rowspan="1" colspan="1">&#35;</th>
    <td rowspan="1" colspan="1">(*)</td>
    <td rowspan="1" colspan="1">&amp;&#35;35;</td>
  </tr>
  <tr>
    <th rowspan="1" colspan="1">&Ocirc;</th>
    <td rowspan="1" colspan="1">&amp;Ocirc;</td>
    <td rowspan="1" colspan="1">&amp;&#35;212;</td>
  </tr>
    <td rowspan="1" colspan="3">(*) pas de code html pour ces caract&egrave;res sp&eacute;ciaux</td>
  </td>
  </tr>
</table>
```

Figure 8: Regroupement de cellules

"tr" que l'on considère comme constituant l'entête du tableau, soit, sa partie descriptive), "tbody" (ensemble de lignes de données du tableau), et "tfoot" (disons, notes de bas de tableau). Ces trois éléments sont les seuls groupes de lignes que l'on peut avoir dans un tableau (un regroupement de lignes permet d'affecter aux éléments et au contenu de ces lignes un comportement commun, mais aussi de dessiner les contours de ces groupes). Un tableau contient 0 ou 1 élément "thead", 0 ou 1 élément "tfoot", et 0 ou n éléments "tbody" (ce qui autorise finalement tous les regroupements de lignes possibles). Noter que *ces trois éléments ne déclarent aucunement des lignes* : ils ne font que déclarer que des lignes décrites par l'élément "tr" appartiennent à un même groupe (et permettent de fait de leur appliquer une certaine mise en forme). Remarque : on n'est nullement obligé de mettre des cellules "th" dans les lignes constituant le groupe d'entête.

thead	:	entête du tableau	0..1
tbody	:	corps du tableau	0..n
tfoot	:	notes de bas de tableau	0..1
Attention	*!*	toujours définir les éléments "thead" et "tfoot" *avant* l(es) élément(s) "tbody"	

Sur l'exemple précédent, on peut considérer que la première ligne est une ligne d'entête ; on peut également vouloir séparer les caractères spéciaux en deux groupes : les caractères accentués, et les autres. Enfin, la dernière ligne consitue une note de bas de tableau. Ces raffinements nous mènent au code XHTML de la figure 9, dont le rendu visuel est le suivant (noter que l'utilisation des éléments de regroupement de lignes ne modifie en rien le rendu visuel) :

```
<table>
  <thead>
    <tr><th>caract&egrave;re</th>code HTML</th><th>Code ASCII</th></tr>
  </thead>
  <tfoot>
    <tr>
      <td colspan="3">(*) pas de code html pour ces caract&egrave;res sp&eacute;ciaux</td>
    </tr>
  </tfoot>
  <tbody>
    <tr><th>&amp;</th><td>&amp;amp;</td><td>&amp;&#35;38;</td></tr>
    <tr><th>&euro;</th><td>&amp;euro;</td><td>&amp;&#35;128;</td></tr>
    <tr><th>&#35;</th><td>(*)</td><td>&amp;&#35;35;</td></tr>
  </tbody>
  <tbody>
    <tr><th>&Ocirc;</th><td>&amp;Ocirc;</td><td>&amp;&#35;212;</td></tr>
    <tr><th>&Ouml;</th><td>&amp;Ouml;</td><td>&amp;&#35;214;</td></tr>
  </tbody>
</table>
```

Figure 9: Regroupement de lignes

caractère	code HTML	code ASCII
&	&amp;	&#38;
€	&euro;	&#128;
Ô	&Ocirc;	&#212;
Ö	&Ouml;	&#214;
(*) pas de code html pour ces caractères spéciaux		

Regroupement et définition de colonnes On peut également déclarer des groupes de colonnes, par l'élément "colgroup". De même que pour les groupes de lignes, les groupes de colonnes servent à structurer (regroupement de contenu) et à formater (largeur, alignement dans les cellules, attribut "style") le tableau, mais pas à en décrire le contenu. Les colonnes d'un tableau étant implicitement définies par ses cellules, on peut regrouper des colonnes avec "colgroup", sans pour autant que les colonnes aient été déclarées par l'élément "col". Pour regrouper une ou des colonnes au sein d'un groupe, il faut écrire :

<colgroup span="n"> ... </colgroup>

Où la valeur n de l'attribut "span" spécifie le nombre des colonnes à regrouper.

Toujours sur la base de l'exemple précédent, on peut considérer que l'on dipose d'un premier groupe de colonnes consitué de la première colonne (groupe d'entête contenant les caractères spéciaux), et d'un second groupe constitué des deux dernières colonnes (groupe de description qui contient les codes des caractères spéciaux). Une telle structuration mènerait au code XHTML de la figure 10. Dans cet exemple, nous avons également utilisé l'élément "col". Celui-ci n'a d'utilité que si l'on souhaite attribuer un comportement (mise en forme) à une ou plusieurs colonnes, sans pour autant que cette colonne ou cet ensemble de colonnes forme un groupe. Son attribut principal est l'attribut "span", qui spécifie le nombre de colonnes concernées. Si par exemple on écrit :

<col span="2" width="150" style="text-align: center;" />

Cela signifie que les *deux* prochaines colonnes sont d'une largeur de 150 pixels et que le contenu des cellules de ces colonnes est aligné au centre. Contrairement à ce que pourrait suggérer l'exemple fourni, on peut aussi déclarer des colonnes à l'extérieur d'un groupe de colonnes (de même que l'on peut spécifier des lignes à l'extérieur d'un groupe de lignes). Attention : l'élément "col" dispose d'une balise unique, conjointement ouvrante et fermante.

```
<table width="1200">
  <colgroup span="1" width="200">
    <col span="1" style="text-align: center;" />
  </colgroup>
  <colgroup span="2" width="1*" style="text-align: left;">
    <col span="1" style="color: red;" />
    <col span="1" style="color: green;" />
  </colgroup>
  <thead>
    ...
</table>
```

Figure 10: Regoupement et définition de colonnes

4.2.2 Mise en forme d'un tableau

Indépendamment des propiétés usuelles de style (que l'on peut affecter par le biais de l'attribut "style" sur tous les éléments d'un tableau), un tableau dispose de propriétés spécifiques de mise en forme :

1. largeur du tableau et de ses colonnes ;
2. bords du tableau et séparateurs entre ses lignes et ses colonnes ;
3. alignement horizontal et vertical du contenu de ses cellules ;
4. espacement entre ses lignes et ses colonnes d'une part, entre le contenu et le bord de ses cellules d'autre part.

Attention : la priorité des attributions de propriété entre éléments diffère selon la propriété considérée. La priorité va toujours du plus petit contenu au plus grand contenant (*ex.* : "td" gagne sur "tbody" ou "colgroup", "tr" gagne sur "thead", "col" gagne sur "colgroup"), mais l'arbitrage entre une spécification de niveau *ligne* (ou *groupe* de lignes) et une spécification de niveau *colonne* (ou *groupe* de colonnes) fera gagner, tantôt la spécification *ligne* (*ex.*, "valign", "style"), tantôt la spécification *colonne* (*ex.*, "align"). Attention également aux prioriétés héritées par une cellule via l'utilisation des attributs "colspan" et "rowspan" : toute cellule d'un regroupement de cellules prend en effet les propriétés de la *première* cellule du regroupement.

Largeur Le tableau définit un bloc particulier, dont la largeur est à définir. On peut le faire de différentes façons :

1. Mode automatique : on ne définit rien, la largeur sera déduite de la taille du contenu.
2. Largeur du tableau : par affectation de l'attribut "width" de l'élément "table". La largeur peut être donnée, soit en valeur absolue (nombre de pixels), soit en pourcentage (valeur alors relative à l'espace horizontal disponible).
3. Largeur des colonnes du tableau : on peut spécifier la largeur des colonnes d'un tableau par le biais des éléments "col" et "colgroup", qui disposent également d'un attribut "width". La largeur d'une colonne peut être donnée :
   (a) en valeur absolue (nombre de pixels, *ex.*, width="150px") ;
   (b) en pourcentage *de la largeur totale du tableau* (*ex.*, width="15%") ;
   (c) par la largeur de son contenu, en utilisant la valeur "0*" (*ie.*, width="0*") ;
   (d) en valeur relative *de la largeur restante du tableau après considération des colonnes dont la largeur aura été spécifiée par l'une des trois manières précédentes* (*ex.*, width="2*").

Dans l'exemple de la figure 10, le tableau dispose d'une largeur de 1200 pixels. Un premier groupe de colonnes est spécifié, dont toutes les colonnes (en l'occurrence, 1 colonne) sont d'une largeur de 200 pixels. Un second groupe de deux colonnes est ensuite spécifié. La largeur des colonnes de ce groupe est de "1*" (valeur relative) : cela signifie que les colonnes se répartissent équitablement l'espace laissé vacant par les colonnes dont la largeur est définie en valeur absolue (nombre de pixels), en pourcentage (pourcentage de la largeur du tableau), en automatique (largeur s'adaptant à celle du contenu). Ici, ces autres colonnes prennent un total de 200 pixels : les deux colonnes se partagent donc un espace de 1200-200 = 1000 pixels, elles seront donc chacune d'une largeur de 500 pixels.

Contours Dans un tableau, deux types de bordures sont à spécifier :

1. les contours du tableau lui-même : attributs “frame” et “border” de “table’ ;
2. les sépérateurs entre lignes et les séparateurs entre colonnes : attribut “rules” de “table’.

L’attribut “frame” définit quels sont les contours du tableau à dessiner :

valeur	:	**void**	**box**	**hsides**	**vsides**
contours	:	aucun (valeur par défaut)	tous	haut et bas	gauche et droit
valeur	:	**above**	**below**	**lhs**	**rhs**
contours	:	haut	bas	gauche	droit

L’attribut “border” donne simplement la largeur (en pixels) de ces contours (*ex.* : border="10"). Attribuer la valeur “0” à l’attribut “border” est équivalent à attribuer la valeur “void” à l’attribut “frame”.

L’attribut “rules” définit quels sont les séparateurs de lignes et les séparateurs de colonnes à dessiner :

valeur	:	**none**	**all**	**groups**	**rows**	**cols**
séparateurs	:	aucun (valeur par défaut)	tous	groupes de lignes et de colonnes	lignes	colonnes

Reprenons notre exemple et considérons à présent le code XHTML de la figure 11. Nous présentons différents rendus visuels en fonction des valeurs données aux attributs “frame” et “rules”.

frameval == void, *rules* == none

caractère	**code HTML**	**code ASCII**
&	&amp;	&#38;
€	&euro;	&#128;
Ô	&Ocirc;	&#212;
Ö	&Ouml;	&#214;

(*) pas de code html pour ces caractères spéciaux

frameval == box, *rules* == none

caractère	**code HTML**	**code ASCII**
&	&amp;	&#38;
€	&euro;	&#128;
Ô	&Ocirc;	&#212;
Ö	&Ouml;	&#214;

(*) pas de code html pour ces caractères spéciaux

frameval == void, *rules* == all

caractère	**code HTML**	**code ASCII**
&	&amp;	&#38;
€	&euro;	&#128;
Ô	&Ocirc;	&#212;
Ö	&Ouml;	&#214;

(*) pas de code html pour ces caractères spéciaux

frameval == box, *rules* == groups

caractère	**code HTML**	**code ASCII**
&	&amp;	&#38;
€	&euro;	&#128;
Ô	&Ocirc;	&#212;
Ö	&Ouml;	&#214;

(*) pas de code html pour ces caractères spéciaux

```
<table frame="frameval" rules="rulesval">
  <colgroup span="1" style="text-align: center;">
  </colgroup>
  <colgroup span="2" width="1*" style="text-align: left;">
  </colgroup>
  <thead>
    <tr><th>caract&egrave;re</th><th>code HTML</th><th>Code ASCII</th></tr>
  </thead>
  <tfoot>
    <tr>
      <td colspan="3">(*) pas de code html pour ces caract&egrave;res sp&eacute;ciaux</td>
    </tr>
  </tfoot>
  <tbody>
    <tr><th>&amp;</th><td>&amp;amp;</td><td>&amp;&#35;38;</td></tr>
    <tr><th>&euro;</th><td>&amp;euro;</td><td>&amp;&#35;128;</td></tr>
    <tr><th>&#35;</th><td>(*)</td><td>&amp;&#35;35;</td></tr>
  </tbody>
  <tbody>
    <tr><th>&Ocirc;</th><td>&amp;Ocirc;</td><td>&amp;&#35;212;</td></tr>
    <tr><th>&Ouml;</th><td>&amp;Ouml;</td><td>&amp;&#35;214;</td></tr>
  </tbody>
</table>
```

Figure 11: Contours

Alignement dans les cellules Tous les éléments “col”, “colgroup”, “thead”, “tfoot”, “tbody”, “tr”, “th”, “td” disposent des attribut “align” (alignement horizontal dans les cellules) et “valign” (alignement vertical dans les cellules). L’affectation d’une valeur à l’un de ces 2 attributs pour l’un de ces éléments sera valable pour *toute cellule* que l’élément contient.

align	:	left	center	right	justify
valign	:	top	middle	bottom	baseline

Espacements On peut définir l’espace *entre deux cellules adjacentes* par l’attribut “cellspacing”, et l’espace *entre le bord d’une cellule et son contenu* par l’attribut “cellpadding”. Ces deux attributs sont des attributs de l’élément “table” (valables donc pour toutes les celulles du tableau), et peuvent prendre comme valeur, une valeur absolue en nombre de pixels, ou une valeur relative en pourcentage de la taille des cellules.

4.2.3 Remarques

Tableau et boîtes Les paramètres d’affichage d’un tableau ne lui sont pas réellement spécifiques, puisque toute boîte graphique associée à un élément XHTML dispose des propriétés :

1. *border* : largeur des quatre bords de la boîte ;
2. *margin* : marge entre chaque bord de la boîte et le bord de la boîte de son élément contenant ou plus proche adjacent (selon les cas) ;
3. *padding* : espace entre chaque bord de la boîte et le bord correspondant de son contenu ;
4. *width* : largeur du contenu de la boîte ;
5. *height* : hauteur du contenu de la boîte.

Ainsi, les paramètres d’affichage que les attributs *width*, *frame*, *rules*, *border*, *cellspacing* et *cellpadding* de “table” permettent de spécifier peuvent également être spécifiés par le biais des propriétés *width*, *border*, *margin* et *padding* des différents éléments constituant le tableau (dont le tableau lui-même). Ces attributs devraient ainsi être amenés à disparaître.

Valeurs relatives On rappelle que les valeurs relatives (hauteurs et largeurs données en pourcentage) sont toujours énoncées vis-à-vis de la boîte de l'élément *père* : la largeur relative d'un tableau sera déduite de la largeur *de contenu* de la boîte définie par l'élément qui le contient ; la largeur relative d'une colonne (et donc, des cellules que cette colonne contient) sera déduite de la largeur de l'élément "table" auquel cette colonne appartient.

Nombre de lignes et de colonnes Il n'y a pas de contrôle sur la cohérence d'un tableau : on peut très bien déclarer 3 colonnes mais sur une ligne, déclarer 15 cellules ; on peut également, par une mauvaise utilisation de l'attribut "rowspan", avoir un nombre de lignes par colonne qui ne soit pas homogène. La gestion d'un tableau demande donc un minimum de rigueur : d'une part, il est bon de rendre le plus explicite possible le nombre de colonnes (en utilisant les éléments "col" et "colgroup") et, d'autre part, il faut vérifier que le nombre de cellules par ligne, ainsi que le nombre de cellules par colonnes, est bien constant (en tenant compte de l'utilisation des attributs "rowspan" et "colspan").

4.3 Un peu plus sur les types de blocs

Pour l'instant, on a surtout discuté des valeurs *block* et *inline* de la propriété "display". Il existe néanmoins d'autres valeurs intéressantes, qui permettent par exemple de faire des cellules de tableau *sans tableau* ou encore, des items de liste *sans liste*. C'est un moyen, d'une part, de gagner plus de souplesse sur la création des boîtes et, d'autre part, de redéfinir si on le souhaite le comportement des objets dédiés ("td", "tr", "li",...) (sans avoir à se préoccuper du comportement des différents types d'objets sur le poste client, mais en repartant d'objets "neutres").

Types de boîtes :

1. list-item : item de liste (c'est un bloc, avec en plus un espace pour la puce).
2. table : tableau.
3. table-caption : légende d'un tableau.
4. table-row: ligne de tableau.
5. table-column : colonne de tableau (de même que pour "col", un élément inséré avec *table-column* pour valeur de sa propriété "display" ne produit aucune boîte, et donc, aucun afffichage).
6. table-cell : cellule de tableau.

Propriétés utiles pour la définition de boîtes :

1. width et height (ex : $x$px pour x pixels) : attention, valable seulement pour les boîtes de type *bloc*
2. margin-left (ex : $x$px pour x pixels) : marge gauche de l'élément, c'est-à-dire, espace entre le bord gauche de l'élément et (en gros) le bord gauche de la boîte de son père, ou le bord droit de l'élément adjacent (selon les cas). Les autres marges sont également définies (*margin-right, margin-top, margin-bottom*), et une valeur donnée à *margin* vaut pour les marges des quatres bords.
3. border-collapse (valeurs : *collapse*, *separate*) : indique si les bordures de deux boîtes adjacentes doivent fusionneer ou, au contraire, être séparées.
4. border (ex : $x$px pour x pixels) : largeur de la bordure d'une boîte (utile pour vérfier les emboîtements !).
5. border-color : couleur de la bordure.