Structure HTML et rendu CSS des balises : bloc et en-ligne

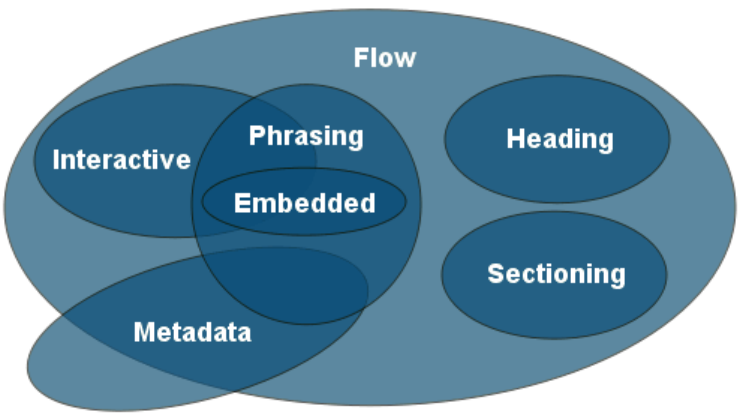
Comprendre la structure HTML et le rendu CSS des balises HTML

Bien souvent sont évoqués des **éléments de type bloc** et des **éléments de type en-ligne**. Il faut savoir que ces désignations sont quelque peu faussées car elles mélangent une partie des spécifications HTML (qui proposent des catégorisations d'éléments) et une partie des spécifications CSS (qui proposent des modèles de rendus).

Historiquement, **HTML** ne proposait que deux catégories d'éléments : **les éléments de niveau block et les éléments de niveau inline**. Cette catégorisation autorise ou non certaines imbrications (par exemple un niveau inline de peut pas contenir de niveau block). En parallèle, les spécifications **CSS** proposent un vaste choix de modes de rendu pour les éléments via la propriété `display`. Parmi les valeurs les plus connues de `display`, l'on retrouve... `block` et `inline`. Forcément, c'est un peu confusant !

En HTML5

La catégorisation est améliorée et modifiée depuis HTML5. Tous les éléments sont regroupés dans des **modèles de contenu** » http://www.w3.org/TR/2011/WD-html5-20110525/content-models.html , ou *Content models,* qui se déclinent dans 7 catégories, dont certaines se recouvrent mutuellement.



Le **flux** (flow) regroupe la plupart des éléments courants, c'est-à-dire les autres sous-modèles cités ci-après, ainsi que le contenu texte simple.

Les **métadonnées** (*metadata*) ne relèvent pas du contenu principal mais participent à la définition des informations gravitant autour de ce dernier, par exemple le titre du document (`<title>`), le style (`<style>`), les relations externes (`<link>`), et les scripts (`<script>`). Il s'agit donc pour la plupart d'éléments *invisibles*.

Le **contenu sectionnant** (*sectioning*), définit les grandes zones du document HTML ou de l'application web : `<article>`, `<aside>`, `<nav>`, `<section>`. Par consensus, les navigateurs ont choisi de conférer à ces éléments un rendu CSS de type **bloc**.

La **titraille** (*heading*) comprend tous les titres hiérarchiques (`<h1>` à `<h6>` et `<hgroup>`), qui sont eux aussi par défaut affichés en **bloc** par les navigateurs.

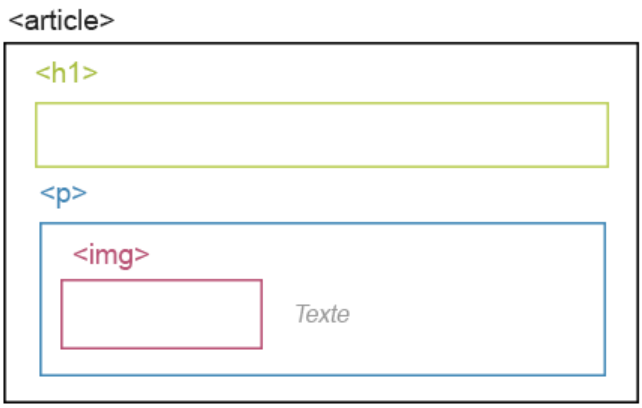
Le contenu de **phrasé** (*phrasing content*) correspond à la plupart des éléments pouvant apparaître dans un flux de texte, et qui sont pour la plupart d'entre eux affichés **en-ligne** (par exemple en CSS `display:inline` ou `display:inline-block`). On peut y remarquer entre autres `<audio>`, `<video>`, `<iframe>`, `<canvas>`, `<img>`, les éléments de formulaire tels que `<input>`, `<textarea>`, `<button>`, `<select>` et le balisage du texte `<strong>`, `<b>`, `<i>`, `<del>`, etc.

Le contenu **embarqué** (*embedded*) est plus spécialisé : `<audio>`, `<canvas>`, `<embed>`, `<iframe>`, `<img>`, `<object>`, `<video>`, `<svg>`, `<math>`.
A titre indicatif, son apparence CSS est définie par des espaces possédant une hauteur et une largeur, souvent affichés avec un type `display:inline-block`.

Le contenu **interactif** est destiné à tout ce qui permet une interaction avec l'utilisateur : les liens, les éléments média, les contrôles de formulaire, bref, tout ce qui peut être piloté au clavier, à la souris ou au doigt.

Emboîtements

Tous les éléments possèdent des règles définissant clairement : dans quel autre élément (parent) ils peuvent être placés, et quels autres éléments (enfants) ils peuvent contenir. Par ailleurs, des balises bien précises sont auto-fermantes, c'est-à-dire des éléments vides et ne doivent -ou ne peuvent- rien contenir du tout ; c'est le cas de `<img />`. Ces règles ne doivent pas être vues comme une contrainte, mais comme le fruit d'une longue réflexion et d'un usage naturel. Il serait par exemple absurde de placer un élément `<input>` dans un `<img>` ou un `<hgroup>` dans un `<select>`.



En respectant ces principes, votre document sera valide et aura toutes les chances d'être bien analysé syntaxiquement, interprété et compris par tous les navigateurs.

De nouvelles exceptions voient le jour toutefois : les liens `<a>` peuvent désormais englober un ou plusieurs éléments de type bloc, contrairement à ce qui était indiqué par les précédentes versions de HTML

En HTML4 et XHTML 1.x

Chaque élément (balise HTML) se caractérise par une double identité :

1. Une *structure HTML* (ou catégorisation) qui n'a "intrinsèquement" aucun rapport avec l'affichage de l'élément.
2. Son *rendu* sur les navigateurs (affichage, positionnement, comportement). Il est défini par défaut selon le bon vouloir de chaque navigateur, et peut être modifiable à l'aide des styles CSS (en utilisant la propriété CSS "display"). Suivant la valeur de cette propriété (les plus fréquentes sont "block" et "inline"), l'élément s'affiche différemment sur le navigateur.

Retenons simplement qu'un élément possède un rendu CSS qui n'a pas forcément de rapport avec sa structure HTML.

Les balises HTML ont **toutes par défaut des propriétés de rendu CSS particulières**. En fait, il existe initialement deux grands groupes principaux de balises : les balises de rendu CSS "bloc" (block) et les balises de rendu CSS "en-ligne" (inline). Ces valeurs de rendu visuel coïncident généralement par défaut avec le groupe d'appartenance HTML.

- Voir la liste HTML4 des éléments de rendu En-ligne » http://htmlhelp.com/reference/html40/inline.html .
  Exemples : SPAN, EM, STRONG, IMG, BR, INPUT, etc.
- Voir la liste HTML4 des éléments de rendu Bloc » http://htmlhelp.com/reference/html40/block.html .
  Exemples : DIV, P, H1...H6, UL, OL, TABLE, PRE, etc.

	Type %block	Type %inline
Exemple	`<p></p>` `<div></div>`	`<i></i>` `<span></span>`
Rendu CSS par défaut	`display:block;`	`display:inline;`

Fonction et emboîtements

La norme HTML4 définit chaque élément par:

- Un groupe d'appartenance HTML dans lequel a été classé cet élément (`%block`, `%inline`, `%flow`, `%heading`,...).
- Une spécification qui indique la fonction de l'élément, c'est-à-dire pour quel type de contenu l'employer.
- Une définition dans la DTD qui liste très précisément les éléments que peut contenir l'élément défini.
- Une liste d'éléments qui peuvent contenir l'élément défini,
- Une liste d'attributs HTML valides pour l'élément défini (ex : `src="..."` pour `<img>`).

La confusion entre la *structure* HTML et le *rendu* CSS vient du fait que les termes se rejoignent (`%block` pour l'un, `display: block` pour l'autre). Notez que les groupes d'appartenance ont été entièrement revus sur la version HTML5 et que les groupes `%inline` et `%block` n'en font plus partie.

Conteneurs et imbrications

En règle générale :

- Un élément de groupe `%block` peut contenir un (ou plusieurs) autres élements `%block` et/ou `%inline`, sauf exceptions » /astuce/lire/55-balises-bloc-et-en-line-les-exceptions.html .
- Une élément du groupe `%inline` ne peut contenir QUE un (ou plusieurs autres) éléments `%inline`.

Exemples d'imbrications :

```
<div><p>Paragraphe 1</p><p>Paragraphe 2</p></div>
```

La balise `<div>` (structure `%block`) englobe les deux paragraphes (structure `%block`). Ceci est autorisé.

Par contre, nous n'aurions pas pu écrire :

```
<span><p>Paragraphe 1</p><p>Paragraphe 2</p></span>
```

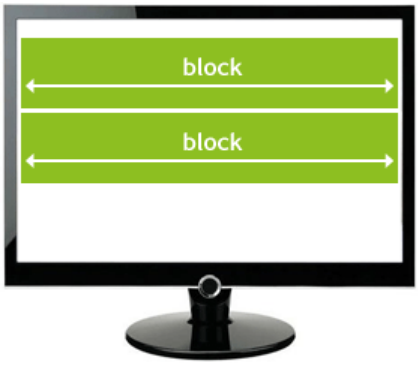
car la balise `<span>` (dont la structure est `%inline`) n'est pas autorisée à contenir des éléments de structure `%block` comme les paragraphes `<p>`.

Rendu CSS : l'aspect par défaut des éléments (concerne HTML4 et HTML5)

Des propriétés CSS d'un élément (`display: block`, `inline`, `list-item`, `inline-block`, `table`, `table-cell`,...) découlent ses spécificités d'affichage, son **positionnement dans le flux**.

block

Les éléments de rendu CSS **block** se placent toujours l'un en dessous de l'autre par défaut (comme un retour chariot). Par exemple: une suite de paragraphes (balise `<p>`). Par ailleurs, un élément "bloc" occupe automatiquement, par défaut, toute la largeur disponible dans son conteneur et peut être dimensionné à l'aide des propriétés telles que `width`, `height`, `min-width`, ou `min-height`,...

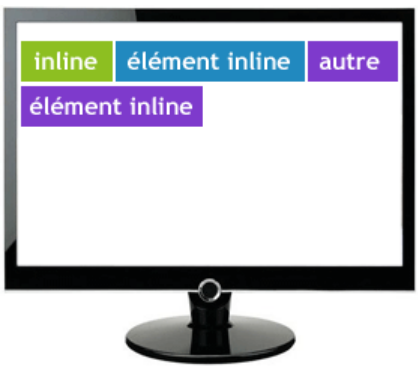



```
<p>Paragraphe 1</p><p>Paragraphe 2</p>
```

Ces deux paragraphes vont s'afficher sur deux lignes, car la balise `<p>` est par défaut un élément de rendu "bloc" : chaque paragraphe va occuper une ligne.

inline

Les éléments de rendu **inline** se placent toujours l'un à côté de l'autre afin de rester dans le texte (par exemple la balise `<strong>`). Par défaut, il n'est pas prévu qu'ils puissent se positionner sur la page (même si cela est possible), ni de leur donner des dimensions (hauteur, largeur, profondeur) : leur taille va être déterminée par le texte ou l'élément qu'ils contiennent. Certaines propriétés de marges peuvent être appliquées aux éléments, comme les marges latérales.

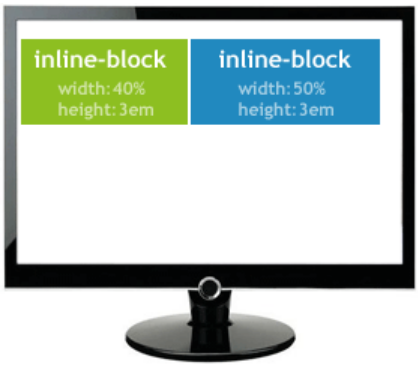



```
<strong>Toto</strong> et <em>moi</em>
```

Ce texte va s'afficher sur une seule ligne (aucun retour à la ligne) car les éléments qui le composent sont de rendu CSS "en-ligne".

inline-block

Les éléments de rendu **inline-block** conservent les mêmes caractéristiques que les "inline", mais peuvent être dimensionnés, par exemple la balise `<input>`.

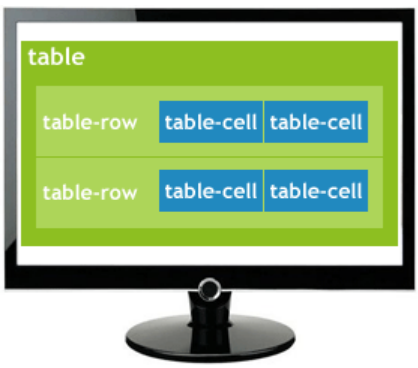


list-item

Les éléments de rendu **list-item** ont un rendu de type "block" mais peuvent bénéficier des propriétés CSS liées aux puces (`list-style`, `list-style-type`, `list-style-image`, `list-style-position` , ...), par exemple la balise `<li>`.

table, table-row, table-cell

Les éléments de rendu **table**, **table-row**, **table-cell** possèdent le même comportement que les éléments de tableaux et cellules (`<table>`, `<tr>`, `<td>`). Ils permettent de centrer les contenus verticalement et d'avoir des hauteurs identiques entre éléments frères, par exemple la balise `<td>`.



Il existe également **inline-table**, **table-row-group**, **table-column**, **table-column-group**, **table-header-group**, **table-footer-group**, et **table-caption**.

autres...

Il existe d'autres valeurs possibles pour la propriété `display`, notamment :

none
: La fameuse valeur qui permet de masquer un élément à l'affichage : aucune boîte n'est générée dans la structure de formatage, il n'y a pas d'influence sur la mise en forme du document.

run-in et compact
: Ces valeurs créent une boîte de bloc ou en-ligne, selon le contexte. Les propriétés s'appliquent aux boîtes compactes ou en

enfilade en fonction de leur statut final (types bloc ou en-ligne).

marker
: Cette valeur précise la qualité de marqueur du contenu généré apparaissant avant ou après une boîte. Elle ne devrait être employée qu'avec les pseudo-éléments `:before` et `:after` liés à des éléments de type bloc. Dans certains cas, cette valeur est interprétée comme 'inline'.

Feuille de style par défaut pour HTML 4

Pour information, voici ci-dessous la feuille de style par défaut proposée par la norme CSS2.1 » http://www.w3.org/TR/CSS21/sample.html par le W3C. Les navigateurs sont invités à l'appliquer :

```
html, address,
blockquote,
body, dd, div,
dl, dt, fieldset, form,
frame, frameset,
h1, h2, h3, h4,
h5, h6, noframes,
ol, p, ul, center,
dir, hr, menu, pre   { display: block }

li              { display: list-item }
head            { display: none }
table           { display: table }
tr              { display: table-row }
thead           { display: table-header-group }
tbody           { display: table-row-group }
tfoot           { display: table-footer-group }
col             { display: table-column }
colgroup        { display: table-column-group }

td, th          { display: table-cell }
caption         { display: table-caption }
input, select   { display: inline-block }
```

Par défaut, la plupart des éléments de groupe HTML `%block` (en fait, tous sauf la balise neutre `<div>`) possèdent un rendu CSS qui s'exprime par des marges internes et externes non nulles. Ce détail est important car ces marges, interprétées différemment suivant les navigateurs, nécessitent parfois d'être annulées afin d'éviter de gros soucis de compatibilité » /article/lire/537-Mon-site-valide-en-XHTML-Strict-n-est-pas-compatible-partout.html qui peuvent être source de divergences de rendu.

Feuille de style par défaut pour HTML5

La grande majorité des éléments HTML5 destinés à être affichés possèdent des styles par défaut. La spécification du balisage HTML5 » http://dev.w3.org/html5/markup/ indique des styles recommandés pour chacun d'entre eux. Libre aux éditeurs de navigateur de suivre ces conseils ou de choisir des styles alternatifs.

Conclusion et rappels

Il est important de retenir qu'un élément se caractérise par :

- Une appartenance à un groupe HTML4 au sein de sa DTD » /article/lire/560-DTD-comment-choisir.html ou un modèle de contenu HTML5.
- Un comportement, ou une apparence, définie par défaut selon le bon vouloir de chaque navigateur, ou modifiable via les styles CSS (propriété display). Il s'agit de son *rendu* graphique sur les navigateurs.

Ces deux notions ne sont pas forcément dépendantes : même si un élément de type *block* (comme par exemple la balise `<p>`)

adopte par défaut un rendu CSS de type `display:block`, il peut très bien adopter un rendu de type `inline` ou autre à l'aide de la propriété CSS `display`. Ceci ne modifiera en rien la structure HTML de cet élément, qui est immuable.