Chapitre 05

CSS
Techniques de mise en page

Objectifs du chapitre :

- Comprendre le modèle de boîte
- Les imbrications de boîtes
- Les zones structurelles de l'HTML 5
- Le positionnement dans le flux courant (flux normal)
- Les autres méthodes de positionnement
- La profondeur : Z-Index
- Flottement et passages à la ligne
- Quel positionnement adopter ?
- Les mises en page liquides
- Les mises en page élastiques
- Les fausses colonnes
- Le centrage horizontal en CSS
- Le centrage vertical en CSS
- Planifier une mise en page
- Organisez votre feuille de style

Le placement des objets d'un document web est la base de toute mise en page. Afin de comprendre ces systèmes de placement, il est nécessaire de bien comprendre trois concepts CSS importants :

- ✓ Le ***modèle de boîte***
- ✓ Le ***positionnement***
- ✓ Le ***flottement***

Ils déterminent la disposition des éléments de la page et sont la base des mises en page CSS.

1. COMPRENDRE LE MODELE DE BOITE

Les sites actuels reposent sur une architecture de boîtes, de types blocs, des textes, des tableaux ou des éléments positionnés. La spécification CSS2.1 définit quatre modèles de positionnement :

- *Block layout* utilisés pour les pages web classiques
- *Inline layout* utilisés pour le texte
- *Table layout* utilisés pour les données tabulaires
- *Positioned layout* utilisé pour le positionnement d'éléments autonomes dans la page

Le CSS3 nous ajoute un modèle de positionnement supplémentaire, le *modèle de boîte flexible* ou *Flexbox.*

1.1. Le modèle de boîte

Chaque balise HTML suit un modèle de positionnement par défaut. Chaque mode de positionnement induit des normes, c'est le modèle de boîte.

A. Boîte de type bloc

Les éléments de type bloc se succèdent verticalement, chaque nouveau bloc se plaçant sous le bloc frère précédent. Les blocs occupent toute la largeur disponible dans leur conteneur.

Prenons par exemple deux blocs différenciés par leur couleur, en HTML :

```
<p class="jaune">Une boîte jaune</p>
<p class="verte">Une boîte verte</p>
```

En CSS :

```
.jaune {
background-color: #ffff00;
}
.verte {
background-color: #00ff00;
}
```

Le résultat:

Une boîte jaune

Une boîte verte

Elles **peuvent être dimensionnées** (width et height) et sont pourvues de marges externes (margin) et internes (padding) qui nous permettent **d'espacer** les boîtes les unes des autres.
Elles ne permettent pas l'alignement vertical du contenu au sein de la boîte et il n'y a pas de lien avec le contexte (les boîtes sont autonomes les unes des autres).

Les principaux éléments créant des boîtes bloc sont : *l'élément div ; Les titres h1, h2, h3, h4, h5, h6 ; Le paragraphe p ; Les listes et éléments de liste ul, ol, li, dl, dd ; Le bloc de citation blockquote* ; …

B. Boîte de type en-ligne

Les éléments « en-ligne » ne définissant pas de boîte autonome se suivent sur la même ligne. Chaque nouvel élément se plaçant directement à la suite du précédent, ils s'affichent côte à côte avec retour à la ligne quand il n'y a plus de place dans le conteneur.

Prenons par exemple deux portions de texte différenciées par leur couleur, en Html :

```
<span class="jaune">Une boîte jaune</span>
<span class="verte">Une boîte verte</span>
```

En CSS :

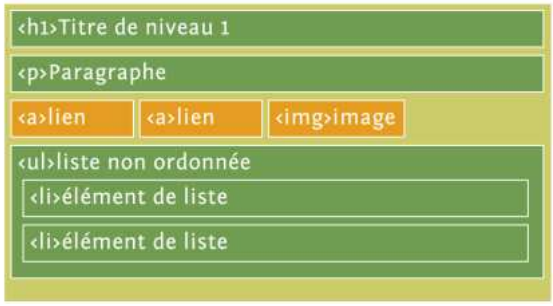
```
.jaune {
  background-color: #ffff00;
}
.verte {
  background-color: #00ff00;
}
```

Le résultat: Une boîte jauneUne boîte verte

Une boîte de type en ligne **ne peut pas être dimensionnée**. Les marges internes ne fonctionnent que sur la droite et la gauche de l'élément et les marges externes ne décalent pas réellement les éléments environnants.
Un **alignement vertical** des éléments inline contenus dans un élément de type block est **possible** grâce à la propriété vertical-align.

Les principaux éléments créant des boîtes en ligne sont : *L'élément span ; Le lien a ; L'image img ; Les emphases em et strong ; …*

Exemple de boîte de type bloc et en-ligne :



C. Boîte de type tabulaire

Le positionnement de ces boîtes dépend du contexte, sans création de boîte autonome. Ainsi, deux boîtes peuvent se positionner côte à côte, leur taille peut être modifiée et un alignement vertical est possible.
Les largeur et hauteur sont fluides et s'adaptent aux boîtes environnantes, deux boîtes côtes à côtes dans un même conteneur auront donc une largeur identique.

D. Eléments hors du flux

Le dernier modèle de boîte concerne les éléments positionnés hors du flux. **Les éléments HTML basculent dans ce modèle lorsque les propriétés CSS `position` ou `float` sont utilisées.**

1.2. Découpage d'une boîte

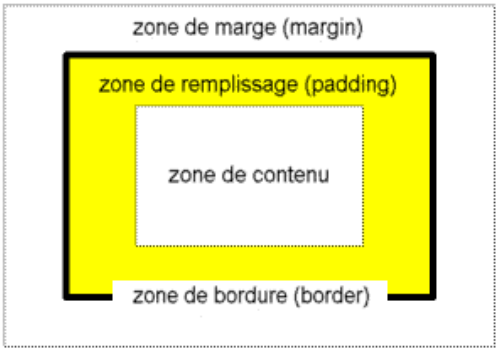
Chaque élément que vous créez dans votre marquage produisant une zone sur votre page html est en fait un agencement de boîtes.

Rappelons tout d'abord qu'une boîte CSS est constituée :

- d'un **contenu** : le texte d'un paragraphe par exemple
- d'un **remplissage** (`padding`) : l'espace entourant le paragraphe
- d'une **bordure** (`border`) : encadrement du paragraphe
- d'une **marge** (`margin`) : espace entourant le tout

Les propriétés CSS permettent de déterminer :

- les dimensions de la zone de contenu, son arrière-plan (image, couleur) et sa couleur d'avant-plan ;
- les dimensions, la couleur et le style des bordures ;
- les dimensions du remplissage ;
- les dimensions de la marge.



Dans chaque cas, il est possible de spécifier des dimensions gauche, droite, haut et bas différentes les unes des autres.

Notez ces **raccourcis intéressant** pour les marges, remplissages ou bordures …

Ecrire un style séparé pour chacun des quatre côtés d'un élément peut être ennuyeux, que vous spécifiiez les marges, le remplissage ou les bordures. Les CSS proposent des raccourcis pour les définir les uns après les autres en une seule déclaration. L'ordre des côtés de la boîte est toujours en haut, à droite, en bas, à gauche (dans le sens des aiguilles d'une montre). Ainsi, pour spécifier les marges d'un élément, au lieu d'écrire

```
{margin-top:5px; margin-right:10px; margin-bottom:12px;
margin-left:8px;}
```

vous écrivez simplement

```
{margin: 5px 10px 12px 8px}
```

On trouve juste un espace entre chacune des quatre valeurs ; inutile d'insérer un délimiteur comme un point ou une virgule.

Il n'est pas primordial de spécifier les quatre éléments ; si vous en manquez un, le système reprend la valeur du côté opposé.

```
{margin: 12px 10px 6px}
```

Dans cet exemple, puisque la dernière valeur manque (à gauche), on utilise la valeur de droite ; la marge de gauche est définie sur 10 pixels.

Dans ce nouvel exemple

```
{margin: 12px 10px}
```

seules les deux premières valeurs (en haut et à droite) sont spécifiées, les valeurs manquantes pour le bas et la gauche sont donc définies sur 12 et 10 pixels respectivement.

Enfin, si on ne fournit qu'une valeur

```
{margin: 12px}
```

les quatre côtés adoptent la même valeur.

Si certaines valeurs doivent être égales à 0, vous pouvez écrire 0 sans fournir de type de valeur, comme ceci

```
{border: 2px 0 0 4px;}
```

1.3. La taille d'une boîte

Le calcul de la taille d'une boîte selon les spécifications CSS2 se fait en ajoutant les valeurs de largeur (`width`), de marges internes (`padding`) et des bordures (`border`). Ainsi, si vous souhaitez une boîte avec un remplissage de

10 pixels et une bordure de 5 pixels de chaque côté mais de 100 pixels de large au total, vous devez fixer la largeur du contenu à 70 pixels.

Voyons le modèle d'une boîte plus en détail :
(Nous l'appliquerons ici à la largeur mais la logique est identique en ce qui concerne la hauteur)

```
#bloc {
      background-color: Silver;
      width: 400px;
      padding: 0 20px;
      border: solid black;
      border-width: 0 6px 0 6px;
      margin: 0 30px;
}
```

En remplissant de 40px une boîte large de 400px, le contenu n'est en rien resserré à 360px mais la boîte est agrandie à 440px. La boîte s'agrandit encore de 12 pixels de par la valeur des 2 bordures gauche et droite.
L'ajout de marge de 30 pixels de chaque côté de la boîte augmente l'espace global occupé par l'élément sans modifier la taille de l'élément lui-même.
Eviter le mode Quirks d'Internet Explorer en ajoutant les bons doctypes. Dans le cas contraire, IExplorer calcule mal la taille des boîtes.

1.4. Box-sizing

La propriété CSS3 **box-sizing** est utilisée pour modifier le modèle de boîte CSS par défaut, **elle permet de modifier la méthode de calcul des dimensions d'une boîte**. Cette propriété est à présent assez bien supportée par la majorité des navigateurs actuels.

Box-sizing accepte 3 valeurs :
Content-box : modèle par défaut
padding-box : les propriétés width et height incluent la marge intérieure, mais pas la bordure ou la marge extérieure.
border-box : les propriétés width et height incluent la marge intérieure et la bordure, mais pas la marge extérieure. C'est le modèle de boîte utilisé par Internet Explorer lorsque le document est en mode de compatibilité (Quirks).

Cette propriété est très pratique lorsqu'on travaille les pourcentages pour fixer la taille des éléments tout en ajoutant des marges internes en px ou en em.

1.5. Minima et Maxima

Une bonne pratique de la conception web accessible à tous est de favoriser autant que possible la fluidité des éléments, or, nous ne pouvons pas dompter la taille des textes que le visiteur va définir selon ses besoins et envies. **Évitez par conséquent de fixer une hauteur à vos boîtes**, sauf s'il s'agit d'un élément dont vous maîtrisez parfaitement les dimensions et le contenu (les images par exemples).

Ces quelques propriétés CSS 2 sont intéressantes à utiliser :

```
min-width : largeur ; /*(en pixels, pourcentage, em...)*/ ;
max-width : largeur maximale;
min-height : hauteur minimale;
max-height : hauteur maximale ;
```

Reconnues depuis Internet Explorer 7, elles ne figent pas les dimensions de l'élément (comme `height` et `width`), mais définissent une valeur minimale ou maximale, quel que soit le contenu présent.

Exemple :

```
body {
  width: 80%;
  min-width: 800px;
  max-width: 1200px;
}
```

Sa largeur est fluide et occupe toujours 80 % de la taille de l'écran, tout en fixant un minimum de 800 pixels et un maximum de 1 200 pixels. **Ces jalons évitent que l'affichage soit dégradé sur les petits écrans ou que le contenu, trop large, soit pénible à lire sur les très grandes surfaces.**

1.6. Notifications sur les marges

A. Les marges par défaut imposées par un navigateur

La plupart des éléments de bloc (paragraphes, titres, listes) possèdent des marges et remplissages par défaut.
Vous pouvez redéfinir ces styles de navigateur en ramenant les propriétés `margin` ou `padding` de l'élément à zéro. Vous pouvez le faire au cas par cas ou pour tous les éléments, à l'aide du sélecteur universel :

```
* {margin:0 ; padding:0}
```

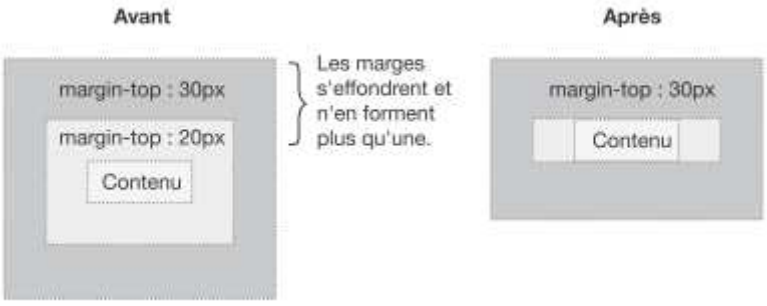
Cette déclaration définit toutes les marges et tous les remplissages à 0 pour qu'il n'y ait pas confusion entre les valeurs spécifiées par les navigateurs et celles que vous indiquez.

B. Le croisement (ou effondrement) des marges

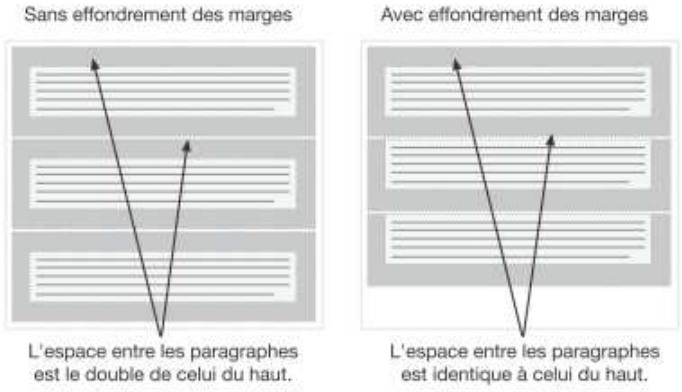
Lorsque deux marges verticales se rencontrent, elles s'effondrent l'une sur l'autre pour n'en former plus qu'une. La hauteur de cette marge fusionnée est égale à la hauteur de la plus grande des deux.
Imaginons 3 paragraphes ayant les styles suivants :

```
p {
      border: 1px;
      border: solid;
      border-color: Black;
      margin-top: 30px;
      margin-bottom: 30px;
      background-color: #E3FFF1;
}
```

Cet effet nous permet de s'assurer que lorsqu'un élément est le premier ou le dernier d'un groupe, il puisse être maintenu éloigné du haut ou du bas de la page de l'élément conteneur (ici, le body).
Lorsqu'un élément est contenu dans un autre élément et qu'il n'existe pas de remplissage ou de bordure séparant les marges, les marges du haut ou du bas sont fusionnées également.



Grâce à l'effondrement des marges, les marges du haut et du bas de chaque paragraphe fusionnent et produisent un espacement égal partout :



L'effondrement des marges ne se produit qu'avec les marges verticales des boîtes de bloc dans le flot normal du document. Les marges entre les boîtes incorporées dans les lignes, flottantes ou positionnées de manière absolue, ne s'effondrent jamais. Les marges internes ("padding") ou des contours ("border") ne génèrent pas le même phénomène.

1.7. Une feuille de style de base (Reset)

Il est préférable de faire appel à une page dite « de reset » plutôt que de remettre à zéro tous les espaces déterminés par le navigateur. Cette page est destinée à redéfinir les styles par défaut des navigateurs (elle corrige ou précise le layout par défaut de ceux-ci) et est généralement appelée **reset.css**.
La feuille de styles « reset » supprime et neutralise l'ensemble hétéroclite des mises en forme par défaut des éléments HTML, afin de créer un layout neutre et uniforme pour tous les navigateurs.

Lecture Web :
Infos détaillées sur le reset :
http://meyerweb.com/eric/tools/css/reset/ et http://yui.yahooapis.com/3.14.0/build/cssreset/cssreset-min.css (pour débuter)
http://html5doctor.com/html-5-reset-stylesheet/
http://www.alsacreations.com/astuce/lire/654-feuille-de-styles-de-base.html
http://knacss.com/
http://www.cssreset.com/scripts/yahoo-css-reset-yui-3/

A. Méthode liée

La feuille de styles de base (ex : reset.css) est appelée de la même manière que d'habitude, **la feuille de styles reset y est déclarée en premier** afin que toutes les suivantes puissent redéfinir les styles réinitialisés comme il se doit :

```
<link rel="stylesheet" type="text/css" media="all" href="css/reset.css" />
<link rel="stylesheet" type="text/css" media="screen" href="css/styles.css" />
```

B. Méthode importée

Une feuille de styles générale (ex : master.css) peut être mise en place pour importer toutes les feuilles de styles utilisées dans le site.

```
@import url("reset.css");
@import url("screen.css");
@import ...
```

Tous les styles pour l'affichage sur écran sont listés dans screen.css.

2. LES ZONES STRUCTURELLES HTML 5

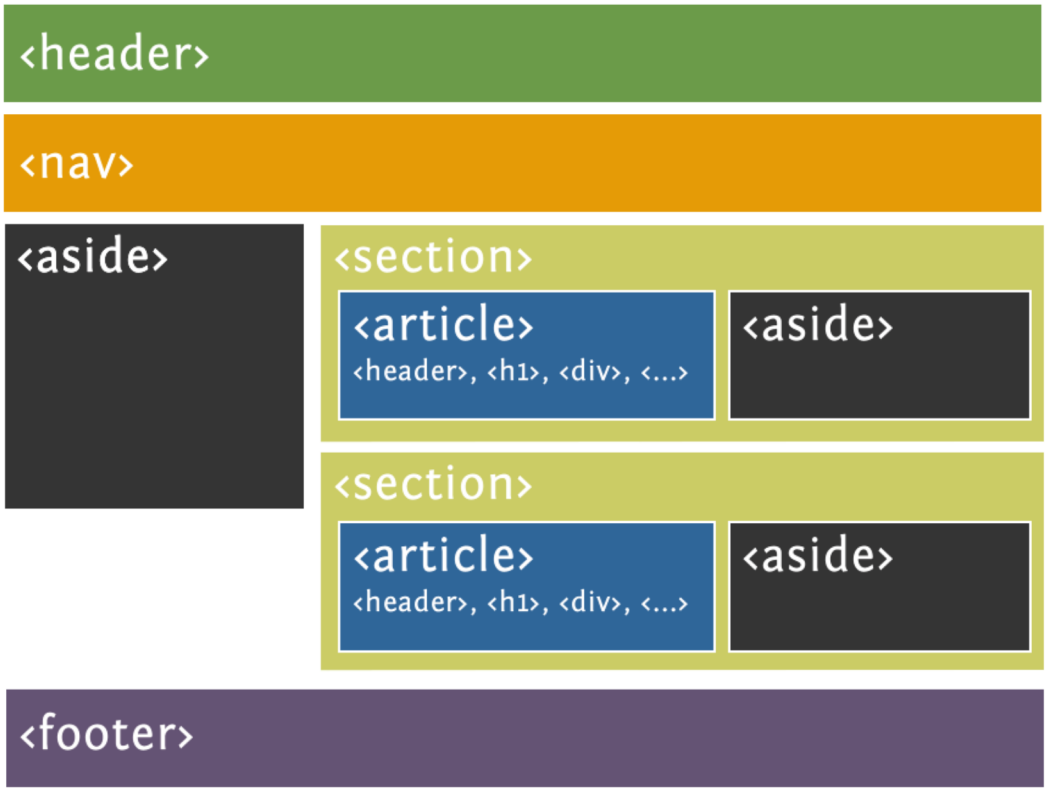
2.1. Les balises HTML5 principales

HTML 5 introduit de nouvelles balises permettant de structurer la page en définissant les différentes sections qui la constitue de manière plus fine qu'en utilisant la balise . Le but est d'apporter de **nouveaux éléments sémantiques dotés de sens**[1] apportant une alternative aux blocs génériques et :
`<article>, <section>, <aside>, <hgroup>, <header>, <footer>, <nav>, <time>, <mark>, <figure>, et <figcaption>.`

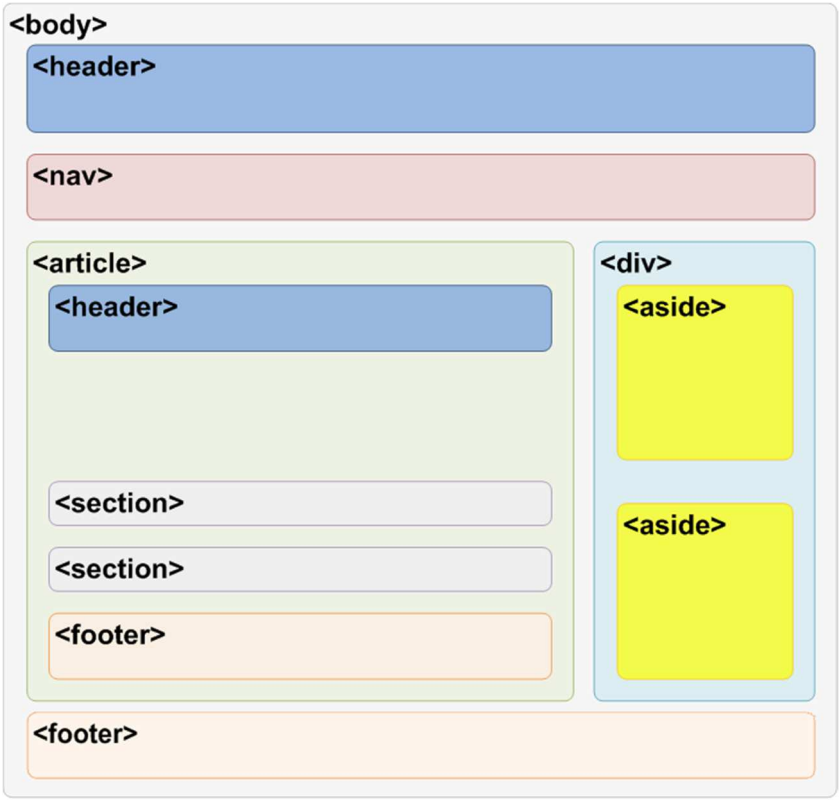
<section>	**Une section permet un regroupement thématique de contenus, généralement avec un titre. Elles peuvent elles-mêmes contenir des sections.** Elles rappellent l'utilisation d'un DIV utilisé auparavant pour englober du contenu sémantique (chapitres, en-têtes et pieds de page). Pour savoir comment utiliser cette balise, posez-vous la question *« est-ce que les contenus sont apparentés ? »*
<header>	**Représente le bloc d'en-tête d'une section ou d'une page.** Destiné à contenir *des éléments d'introduction* (les habituels en-têtes de page – logo, slogan, titre, navigation) ou l'entrée d'une section, d'un article ou une table des matières.
<nav>	**Regroupe les liens de navigation considérés comme majeurs ou jugés suffisamment pertinents (souvent la navigation principale).** Dans la plupart des cas, elle se situera dans le <header>. Toutes les listes de liens ne doit pas être contenues dans une balise <nav>, réservez-le à la navigation à l'intérieur du site.
<article>	**Désigne une portion du document potentiellement autonome** (pouvant être traitée indépendamment du site) **dans le sens où elle pourrait être reprise ou réutilisée – comprise hors contexte** (comme un article de journal, de blog ou de forum)**.** Pour savoir comment utiliser cette balise, posez-vous la question *« ce contenu pourrait-il être diffusé seul ou être syndiqué (flux rss) ? »*
<aside>	**Représente une portion de contenu de type secondaire, contextuelle, directement ou indirectement liée aux éléments qui l'entourent** (relatif à une section du document ou au document lui-même). Exemples : barre latérale contenant des billets similaires, un nuage de tags, une citation en exergue … Pour savoir comment utiliser cette balise, posez-vous la question *« ce contenu pourrait-il être soustrait sans affecter le sens du contenu auquel il se rapporte ? »*
<footer>	**Regroupe les contenus du pied d'une section ou d'un document (pied de page)** et est destiné à recueillir les informations concernant l'auteur, les mentions légales, les sources, etc.
<figure>	**Représente une unité de contenu autonome, généralement référencé comme étant une unité indépendante du contenu du document.** *S'emploie combiné à <figcaption> pour structurer des illustrations, photos, vidéos, diagrammes et portions de codes. <figcaption> désignera la légende de la figure, elle est optionnelle.*

[1] Les noms de ces balises résultent d'une étude Google de 2005 reprenant les noms les plus couramment utilisés pour définir les différentes sections des pages web.

Exemple de structure [1] :



Exemple de structure[2] :



[1] Source : *CSS avancées, Vers HTML 5 et CSS 3* - Raphaël Goetter, Eyrolles
[2] http://www.blog-nouvelles-technologies.fr/archives/5166/la-structure-dune-page-html5/

Ces nouvelles balises HTML5 **offrent aux navigateurs web une toute nouvelle façon d'appréhender votre contenu**. *Dans le futur, le rôle confié à ces éléments pourra donc s'avérer important.*

2.2. Compatibilité de l'HTML 5

a) Utiliser les nouveaux éléments structurels de l'HTML5

Si vous souhaitez utiliser **les nouveaux éléments structurels de l'HTML5** dès aujourd'hui, rien ne vous en empêche car la plupart des navigateurs supportent aujourd'hui ces éléments. Les plus anciens vous permettent de les utiliser et de les styler (*ils les interprètent comme des éléments génériques tels et)*.

Appliquer des styles aux éléments structurels de l'HTML5

Les navigateurs anciens n'appliqueront pas de style par défaut aux nouveaux éléments structurels, il faudra donc, au minimum, déclarer que les nouveaux éléments structurels doivent imposer un saut de ligne :

```
section, article, header, footer, nav, aside {
display: block;
}
```

.

Internet Explorer, jusqu'à la version 9, refuse catégoriquement de reconnaître les nouveaux éléments, à moins qu'un exemplaire de chaque élément n'ait été créé au préalable à l'aide de JavaScript, comme ceci :

```
document.createElement('section');
```

Remy Sharp, programmeur JavaScript, a écrit un petit script très pratique qui génère tous les nouveaux éléments HTML5.

Chargez ce script avec un commentaire conditionnel afin qu'il ne soit utilisé que par Internet Explorer :

```
<!--[if lt IE 9]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

Vous pouvez maintenant styler les nouveaux éléments à votre aise.

2.3. L'attribut "role"

L'attribut "role" permet d'ajouter du contexte aux éléments (de la sémantique).

Il est ainsi possible pour les agents utilisateurs (navigateurs, moteurs de recherche) et pour les applications accessibles riches (ARIA) d'exploiter ces informations complémentaires.

HTML5 définit une liste de valeurs pour l'attribut "role", les principales étant[1] :

main : définit le contenu principal d'un document

secondary : représente une section unique et secondaire du document, tel que l'heure, la météo ou autre module de ce genre

navigation : définit le menu de navigation du document. Typique de liens vers d'autres pages du site ou vers d'autres endroits de la page

banner : située en haut de page, elle contient habituellement le logo de la société, les éléments publicitaires,...

contentinfo : informations à propos du contenu de la page : notes, copyrights, mentions légales, ...

definition : représente la définition d'un élément

note : correspond à une note entre parenthèse ou de bas de page

seealso : indique que l'élément contient des données contextuelles au contenu principal de la page

search : la section de recherche d'une page. Typiquement un formulaire de recherche

[1] Source : http://www.w3.org/TR/xhtml2/mod-roleAttribute.html#s_roleAttributemodule traduit par http://www.alsacreations.com/tuto/lire/1329-osez-html5-et-css3.html