

La balise <body>

C'est à l'intérieur de <body></body> que sont utilisées toutes les balises qui vont servir à afficher quelque chose à l'écran.

Dans les grandes lignes, nous allons trouver toutes les balises permettant de **structurer la page**, c'est-à-dire de la découper en créant des cases, dans lesquelles il sera possible de mettre le menu du site, ou une zone d'image et une légende en dessous, ou un bloc pour un formulaire.

En fait, la structure des sites a longtemps été dessinée uniquement avec des tableaux. C'était des tableaux très simples : il suffisait de mettre dans telle cellule le menu, dans telle autre la liste des articles proposés sur le site, une cellule tout en haut pour une publicité par exemple...

Cette méthode avec les tableaux <table></table> (cf. chapitre Mise en page HTML et CSS - section Les tableaux pour plus de détails sur les tableaux) est relativement simple et peut être intéressante pour afficher des données, comme cela serait fait avec un tableur. Le souci est qu'aujourd'hui il devra également s'afficher correctement sur les smartphones dont les écrans sont plus petits que ceux des ordinateurs. Il ne sera donc pas possible d'afficher le site de la même façon. Autrement dit, il va falloir organiser l'affichage pour que cela fonctionne sur tous les écrans. Il faudra tenir compte également du mode paysage (horizontal) ou portrait (vertical).

Il existe des balises qui sont des blocs (comme un rectangle) et qui peuvent prendre n'importe quelle taille. Ils peuvent être affichés à tous les endroits voulus dans la page, en étant devant ou derrière un autre bloc... Ces blocs sont très pratiques à manipuler et permettent des choses que les tableaux <table> ne permettent pas.

Pour donner quelques balises correspondant à ces blocs, il y a par exemple les balises <div>, <section>, <article>, <nav>, <aside>, <header>, <footer>...

Une fois que toutes les zones dans lesquelles sera affiché du texte ou des images, vidéos, infos... sont définies, le contenu pourra être écrit.

Avant de fabriquer une page web, il faut avoir une bonne idée du résultat final désiré. L'étape de création de la structure, l'organisation des blocs, les dimensions, les positions est sûrement la plus importante et celle qui doit demander le plus d'attention, car tout le reste du site va reposer sur cette structure. Et s'il est nécessaire de changer par la suite l'affichage, la disposition pour une raison ou une autre, dans certains cas cela se fera assez rapidement, et dans d'autres cas cela pourra demander énormément d'efforts, un peu comme si une personne décidait de changer la place de la cave dans la maison : cela reste possible, mais si on y avait pensé à la construction, cela aurait été beaucoup plus simple et rapide à réaliser.

Une fois la structure en place, il « suffira » de la remplir avec les images et textes souhaités. Pour cela, le webmaster a accès à différentes balises qui lui permettent de créer des listes à puces (*Unordered List* : liste non ordonnée) ou ordonnée (*Ordered List*), ou encore d'autres balises permettant d'écrire un titre en gros et gras <h1></h1>.

Il y a également une balise pour créer un paragraphe <p>, et à tout cela viendront s'ajouter des styles pour finaliser l'affichage. Tous ces points seront vus en détail au fur et à mesure.

Si vous avez survécu à toute cette théorie, la pratique arrive pour vous soulager un peu avec un exemple concret de mise en page.

1. Méthode et balises pour structurer une page

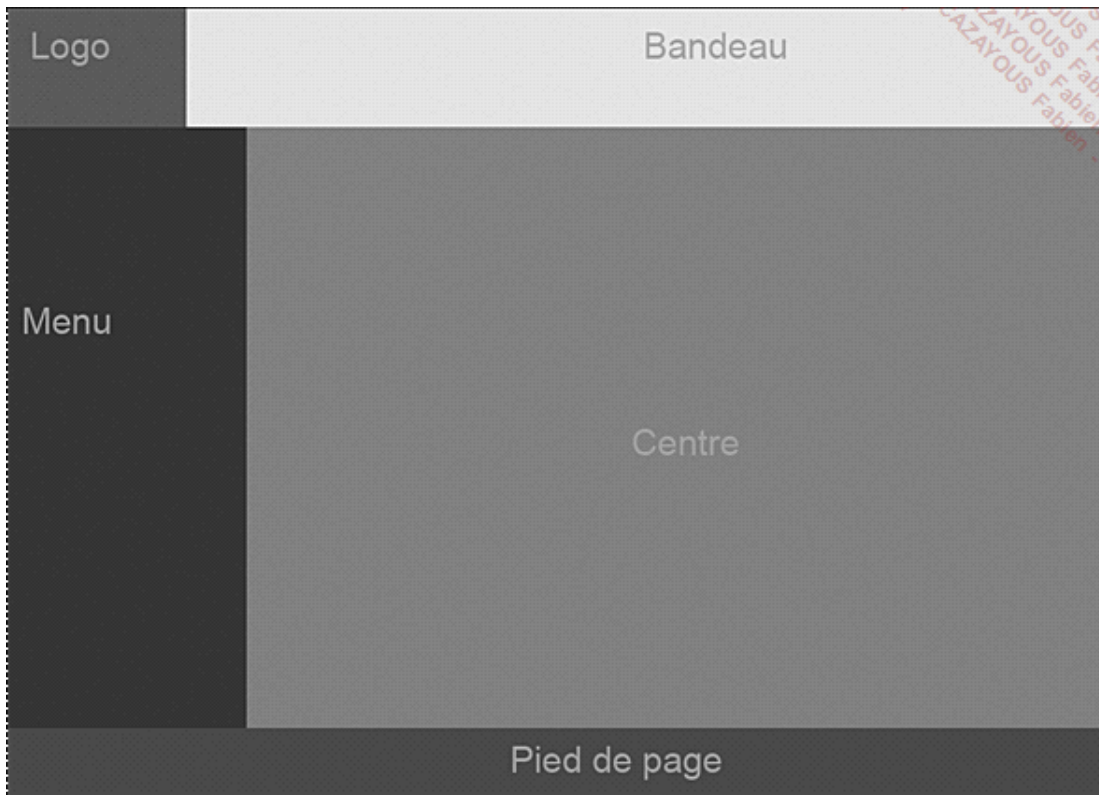
Une des balises qu'il est intéressant de savoir manipuler est la balise <div>. Elle a été la première du genre pour ensuite être imitée par les autres balises <section>, <article>, <nav>, <aside>, <header> et <footer>

qui fonctionnent comme le `<div>`, mais qui ont en plus une identité qui permet de savoir un peu ce qu'elles contiennent.

Par exemple, la balise `<footer>`, de par son nom, indique qu'elle est le pied de quelque chose. Par pied, il faut comprendre zone la plus basse. S'il s'agit d'une page complète, le `<footer>` sera le bas de page. S'il s'agit d'un article, le `<footer>` sera le pied de l'article qui pourra contenir par exemple des informations sur l'auteur de l'article.

Ces balises permettent de définir des blocs, ou rectangles ; des zones dans lesquelles il est possible de mettre tout ce qui doit être affiché ou même d'inclure d'autres blocs.

Après avoir dessiné au brouillon l'allure que devrait avoir le site Internet, il est possible de le définir dans la page HTML à l'aide de balises `<div>`.



Supposons qu'on doive créer une page HTML avec un en-tête de page (ou *header*), qui serait donc le haut de la page et qui pourrait contenir un top avec d'un côté le logo de la société et pour la place restante une ou plusieurs phrases d'accroche représentant la société ou un bandeau publicitaire.

En dessous, il pourrait y avoir sur la gauche un menu vertical et à côté de ce menu la zone principale qui irait du menu jusqu'à la droite de la page.

Il resterait de la place en bas de la page, pour y mettre un `<footer>` avec un accès aux informations légales et autres liens utiles.

Voyons ce que donne le code de la page qui vient d'être décrite et que vous pouvez retrouver en ouvrant le fichier **3_5_1_Structure.html**.

```
<!DOCTYPE html>
<html>
  <head>
```

```

    <title>Structure de la page</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
  </head>

  <body>
    <header id='topPage'>
      <div class='logo'></div>
      <div class='bandeau'></div>
    </header>

    <div id='mainPage'>
      <nav class='menuGauche'></nav>
      <div class='centre'></div>
    </div>

    <footer id='basPage'></footer>
  </body>

</html>

```

Nous voyons, en regardant le code, que nous avons trois parties dans le corps de la page (dans <body>) : le <header> et le <footer> en haut et en bas, et au centre un div ayant pour identifiant #mainPage.

La structure de notre page est définie et il faudra ajouter du contenu, par exemple une image dans le div qui a la classe logo ou encore un menu pour naviguer dans notre site et qui sera dans la balise <nav>.

Mais avant cela, il va falloir apporter des informations de dimensions pour dire quelle taille fait notre page en largeur ou quelle est la hauteur que doit faire le #topPage pour pouvoir afficher le logo.

C'est au niveau du CSS que tout cela va se régler. Nous allons dans un premier temps définir les dimensions des différents éléments, et également, pour s'y retrouver un peu, **mettre des couleurs d'arrière-plan**, qui pourront disparaître ensuite mais qui nous permettront de voir le bloc et de vérifier s'il est bien positionné ou non.

Nous avons vu que pour créer un style d'ID, il faut au niveau du CSS écrire #leNomDeLiD. Nous allons donc pouvoir commencer avec l'identifiant topPage.

Puisqu'il est écrit en premier dans la partie <body> de notre page, il sera situé en haut de la page.



Il est possible, avec ces types de balises de bloc, d'écrire la balise à un endroit dans le code, au début par exemple, et de la positionner à un autre endroit. Des règles de style comme « position » accompagnées de « left » et/ou « top » permettent de la placer où on veut. Voyez la partie CSS pour plus de détails.

Pour pouvoir mettre en place le CSS, il nous faut créer un fichier CSS, que nous pouvons sauvegarder avec le nom « 3_5_1_structure.css », et ensuite ajouter dans la balise <head> de notre page un lien vers la feuille de style.

Le <head> aura alors cette allure :

```

<head>
  <title>Structure de la page</title>
  <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">

```

```
<link href="../../css/3_5_1_structure.css" type="text/css"
rel="stylesheet" media="screen"/>
</head>
```

Et nous aurons au niveau CSS quelque chose qui ressemble à ceci :

```
#topPage {
    width : 100% ;
    height : 100px ;
    background-color : #00FF00 ;
}
```

Nous avons ainsi défini le bloc qui a pour ID `topPage`.

Un ID étant unique, seul un bloc peut avoir l'ID `topPage`. Ce bloc aura une largeur de 100%. Mais 100% de quoi ?

En fait, lorsque l'unité utilisée est le pourcentage, il s'agit du pourcentage par rapport à l'endroit dans lequel le bloc se trouve. Ici, le `<header>` qui a pour ID `topPage` est situé directement à l'intérieur de la balise `<body>`. Autrement dit, il est directement dans la page et non à l'intérieur d'un autre bloc comme `logo` par exemple. La ligne suivante :

```
width : 100%;
```

fait donc référence à la largeur du bloc. Il peut prendre toute la largeur (100%), et aura donc la même largeur que la page. Si la fenêtre du navigateur web diminue en largeur, le bloc `<header>` se réduira proportionnellement à la largeur de la page.

La hauteur du bloc étant en px (en pixels), cette valeur ne changera donc pas de façon dynamique comme le fait une dimension exprimée en %.

Pour finir, l'instruction :

```
background-color : #00FF00 ;
```

va définir la couleur d'arrière-plan en vert (le rouge est à 0 et le bleu aussi). Cela ne va pas donner un résultat très joli pour le moment, mais le bloc `<header>` va bien ressortir et il sera plus simple de vérifier que tous les blocs sont bien à leur place.

Le `#basPage` sera quasiment identique au `#topPage`, si ce n'est la hauteur. Nous pourrions donc avoir quelque chose du genre :

```
#basPage {
    width : 100%;
    height : 60px;
    background-color : #0000FF;
}
```

Nous aurons un pied de page qui fera toute la largeur de la page, pour une hauteur fixée à 60 pixels et une couleur d'arrière-plan bleue.

Il reste maintenant la partie centrale.

Supposons que la partie centrale nécessite une hauteur de 500 pixels pour afficher le détail :

```
#mainPage {  
    width : 100%;  
    height : 500px;  
    background-color : #FF0000;  
}
```

Une fois ces étapes mises en place, il est temps de s'intéresser au niveau inférieur, à savoir le contenu du top, du centre et du bas de page.

Cela va fonctionner de la même façon avec des dimensions et une couleur d'arrière-plan pour s'y retrouver. Il sera expliqué un peu plus loin les notions de marges extérieures (margin) ou marges intérieures (padding) indispensables à un affichage esthétique et aéré. Le CSS final pourrait ressembler à ceci :

```
#topPage {  
    width : 100%;  
    height : 100px;  
    background-color : #00FF00;  
}  
.logo {  
    width:150px;  
    height:100%;  
    background-color: #006600;  
}  
.bandeau {  
    width:100%;  
    height:100%;  
    background-color: #000066;  
}  
#mainPage {  
    width : 100%;  
    height: 500px;  
    background-color : #FF0000;  
}  
.menuGauche {  
    width : 200px;  
    height: 100%;  
    background-color : #660000;  
}  
.centre {  
    width : 100%;  
    height: 100%;  
    background-color : #FF00FF;  
}  
#basPage {  
    width : 100%;
```

```
height : 60px;
background-color : #0000FF;
}
```

Lorsque cette page sera affichée, il y aura deux éléments qui ne vont pas apparaître : `.bandeau` et `.centre`.

Ces deux éléments devraient normalement être affichés à droite d'un autre élément : le `.bandeau` à la droite du `.logo` et le `.centre` à la droite du `.menuGauche`.

Le souci est dans ce cas qu'un `<div>` ou la balise `<nav>` pour le menu, même s'ils ne font pas toute la largeur de la page, ne permettent pas que quelque chose d'autre vienne s'afficher à côté d'eux.

Par défaut, ces balises sont conçues pour utiliser toute la largeur qui est à leur disposition. Le flux de texte est tel que ce qui suit un `<div>` est obligatoirement au-dessous du `<div>`. Donc par défaut, même si le `<div>` ne fait que 50% de la largeur de la page, il ne permet pas que quoi que ce soit se mette à la même hauteur que lui.

float

Heureusement il est possible de dire au `<div>`, en utilisant des styles, de « flotter » sur le côté. Par flotter, on entend que le bloc se calera sur un côté comme il le fait par défaut, mais permettant à d'autres éléments, s'il y a la place, de se mettre juste à côté d'eux.

Ainsi, la propriété `float` ajoutée au style réglera le problème.

```
.logo {
  width:150px;
  height:100%;
  background-color: #006600
  float :left;
}
```

Donc, grâce au `float` appliqué à la classe `.logo`, l'élément qui suivra le `.logo`, dans cet exemple le `.bandeau`, se positionnera juste à côté du logo.

Il faudra faire de même pour le `.menuGauche` de façon à ce que l'élément qui a pour style de classe `.centre` se positionne à côté du menu.


Au final, le CSS ressemblera à ceci :

```
#topPage {
  width : 100%;
  height : 100px;
  background-color : #00FF00;
}
.logo {
  width:150px;
  height:100%;
  background-color: #006600;
  float :left;
}
.bandeau {
```

```

width:100%;
height:100%;
background-color: #000066;
}
#mainPage {
width : 100%;
height: 500px;
background-color : #FF0000;
}
.menuGauche {
width : 200px;
height: 100%;
background-color : #660000;
float :left;
}
.centre {
width : 100%;
height: 100%;
background-color : #FF00FF;
}
#basPage {
width : 100%;
height : 60px;
background-color : #0000FF;
}

```

 Tout cela permet de remarquer qu'au final, le plus gros travail de mise en page est fait au niveau du CSS.

Un des avantages de travailler avec des `<div>` (ou toutes les balises citées précédemment comme `<header>`, `<nav>`...) est justement que l'essentiel de leur paramétrage est fait dans le CSS et il sera alors possible de programmer le CSS pour qu'il s'adapte en fonction de l'écran ; c'est ce qui est appelé le *responsive design*.

2. Le texte dans la page HTML

Une fois tous les blocs positionnés, il est alors possible de mettre du texte dans chaque bloc.

Le choix de la police, de la taille et de la couleur du texte se fait via un ou plusieurs styles. Et il est possible d'appliquer un style pour le titre, un autre pour le texte et encore un autre pour faire ressortir un mot important.

Pour cela, nous avons principalement trois balises à utiliser.

Paragraphe `<p>`

La balise `<p>`, pour paragraphe, a la même attitude que le `<div>`, dans le sens où elle prendra par défaut toute la largeur de la zone dans laquelle elle se trouve. Si elle est utilisée au milieu d'un texte, l'affichage ira directement à la ligne, après un saut de ligne.

Exactement de la même façon qu'après avoir appuyé sur la touche [Entrée] dans un logiciel de traitement de texte, le curseur va à la ligne et crée un espace au-dessus avec le paragraphe précédent.

Niveau code HTML, cela pourrait donner quelque chose du genre :

```
...
<p class='titre'>Les balises HTML5</p>
<p>Il y a de nombreuses nouvelles balises et elles permettent de
faire ... </p>
...
```

La première balise `<p>` récupère le style de classe `titre`, qui écrira le texte plus gros et en gras. La seconde balise n'a pas de style, si bien qu'elle aura le style par défaut de la page, autrement dit le style appliqué à la balise `<body>`.

Supposons maintenant que le mot « nouvelles » dans la seconde balise `<p>` doive être en rouge pour le faire ressortir. Pour appliquer un style, il nous faut une balise, comme cela a été fait pour le titre. Le souci avec la balise `<p>` est que si elle est ajoutée autour du mot « nouvelles », elle va créer un nouveau paragraphe et un saut de ligne, alors que ce qui est souhaité est simplement de mettre en couleur le texte, sans changer la police, la taille ou la position.

Il existe pour cela une balise qui est parfaite, puisqu'elle ne modifie pas l'affichage et ne produit pas de retour à la ligne. Un mot encadré par la balise `` ne sera pas modifié, sinon par le style du ``.

Si notre CSS contient le style :

```
.rouge {
    color :red;
}
```

il sera alors possible d'écrire :

```
...
<p class='titre'>Les balises HTML5</p>
<p>Il y a de nombreuses <span class='rouge'>nouvelles</span>
balises et elles permettent de faire ... </p>
...
```

Dans ce cas, le mot « nouvelles » est encadré par le `span` qui utilise le style `rouge`. Le mot « nouvelles » s'écrira donc en rouge.

**
**

Une dernière balise bien pratique lorsque l'on manipule du texte est la balise `
` (*broken row*, casser la ligne). Cette balise va permettre de faire un retour à la ligne sans changer de paragraphe, et donc sans saut de ligne. C'est l'équivalent de [Shift][Entrée] dans un logiciel de traitement de texte.



Remarquez au passage que la balise `
` fonctionne seule, il n'y a pas deux balises `
` et `</br>`. Beaucoup de balises fonctionnent comme des poupées russes et ont un début et une fin et il est possible de mettre du contenu entre les balises. Mais dans le cas du `
`, il est à la fois le début et la fin.

3. Les caractères spéciaux

Il existe une façon de coder certains caractères pour être sûr qu'ils s'afficheront correctement sur tous les navigateurs. Le terme utilisé pour décrire ces caractères est **entité**.

À l'adresse http://alexandre.alapetite.fr/doc-alex/alx_special.html, vous pourrez trouver facilement la liste complète des entités existantes, mais seules quelques-unes sont indispensables.

L'écriture d'une entité se fait toujours en commençant par une esperluette (& ou « et commercial »), suivie par le nom de l'entité puis d'un point-virgule « ; » pour finir.

` ` signifie *non breakable space* (espace insécable), c'est-à-dire un espace qui sera considéré comme une lettre et ne permettra pas un retour à la ligne entre les deux mots qu'il sépare. S'il faut afficher « Monsieur De Labas », il serait préférable d'éviter de couper le nom de famille et d'avoir « De » en fin de ligne et « Labas » à la ligne suivante. Pour cela, on utilisera l'entité ` ` de la façon suivante :

```
<p> ... Monsieur De&nbsp;Labas ... </p>
```

L'entité ainsi placée empêchera de couper le nom de famille. D'où son nom, insécable.

€

Cette entité permet d'afficher le symbole € (euro), qui est un caractère spécial et qui, à ce titre, peut ne pas s'afficher correctement.

< et >

Ces deux entités, `<` (*lesser than*, plus petit que) et `>` (*greater than*, plus grand que) vont être bien pratiques pour afficher un signe inférieur « < » ou un signe supérieur « > ». L'idée ici est assez simple. Avec le code HTML, ces deux symboles « < » et « > » sont beaucoup utilisés pour écrire les balises, et le problème qui risque d'être rencontré est une confusion entre un simple signe inférieur et le début d'une balise. Donc pour afficher un signe inférieur, il est parfois préférable d'utiliser l'entité `<`.

&

Cette entité permet d'afficher l'esperluette (&), là encore pour éviter une confusion avec une entité qui commence avec le symbole &.

Caractères accentués

Si, pour une raison ou une autre, la page HTML n'est pas définie avec le charset UTF-8, il est possible que des problèmes surviennent pour l'affichage des accents. Dans ce cas, des entités pour les caractères accentués existent.

`à` affiche la lettre « a » avec un accent grave : à.

â correspond au « a » avec un accent circonflexe : â.

L'entité &ecute; affiche le « e » avec un accent aigu, é ou &egave; pour le è.

Et ainsi de suite pour les autres voyelles.

Autres entités

Pour finir, quelques dernières entités utiles :

© permet d'afficher le symbole du copyright : ©.

" pour afficher des guillemets : ".

œ pour afficher l'e dans l'o : œ.