Structure de page et balises structurantes

I. Structure de page

On a coutume d'organiser les pages HTML de base en distinguant un en-tête de page, un corps de page et un pied de page. À ces trois éléments, qui ne diffèrent guère de ce que l'on peut constater dans l'impression papier, il faut en ajouter un dernier qui est propre à l'univers des sites web, à savoir la navigation.

Avant le HTML5, on utilisait des **div** avec un identifiant pour différencier ces grandes parties afin de disposer d'élément de blocs. On a également pour tradition de les envelopper dans une **div** qui vient redoubler la balise **body** et que l'on trouve souvent, nommée **#wrapper**. Voici le code d'une telle page :

```
<div id="wrapper">
    <div id="header">
        <p> header</p>
    </div>
    <div id="nav">
        <p>menu</p>
    </div>
    <div id="main">
        <p>contenu</p>
    </div>
    <div id="footer">
        <p> footer </p>
    </div>
</div>
```

Fig. 1 Structure1 © DR

Notez qu'il est également courant que le menu fasse partie du **header**. Il faut parler rapidement de ces éléments et en décrire le fonctionnement.

Header : l'en-tête de page reste en général la même sur toutes les pages du site même si on voit des **header** qui changent en fonction des rubriques du site, avec un code couleur différent par exemple.

Nav : il s'agit bien entendu de la navigation qui reste également quasiment identique d'une page à l'autre.

Main : c'est le principal élément de la page qui se différenciera, aussi bien en termes de présentation que de contenu d'une page à l'autre. Autre identifiant couramment utilisé : **#contenu**.

Footer : on trouve enfin un élément qui reste le même à chaque page et comporte souvent les éléments tels que le copyright, un lien vers les mentions légales ou encore un *sitemap*.

On va ajouter un peu de mise en forme CSS pour se rendre compte plus précisément du résultat.

Commençons par le **wrapper**. Il va nous servir de boîte principale dans laquelle les autres éléments vont s'empiler et prendre toute la largeur que leur autorise leur parent. On lui indique donc une largeur de 960px (par exemple) et l'on va le centrer grâce à **margin: 0 auto**. C'est ce qui permet de centrer un élément horizontalement, en effet, les marges gauche et droite, étant calculées automatiquement, elles vont prendre la même valeur, quelle que soit la largeur de l'écran. On ajoute également un petit **padding** gauche et droite. Soit :

```
#wrapper  {
    padding:0px 10px;
    margin:0 auto;
    width:960px;
}
```

Fig. 2 Structure2 © DR

À partir de là, il suffit de donner une hauteur (qu'on préfère minimale, min-height), une couleur de fond et une bordure. Soit :

```css
#header{
    min-height:250px;
    background: #F11F33;
    text-align: center;
    border: 1px solid #000;
}

#nav {
    min-height:50px;
    background:#00DD11;
    text-align: center;
    border: 1px solid #000;
}

 #main{
    min-height:450px;
    background: #CD55FF;
    text-align: center;
    border: 1px solid #000;
 }

#footer{
    min-height:50px;
    background: #FDD131;
    text-align: center;
    border: 1px solid #000;
}
```

Fig.3 Structure3 © DR

On a donné une couleur de fond différente à chaque élément pour les différencier, la hauteur de **nav** et **footer** étant plus petite que les autres blocs afin de se conformer avec ce qu'il se fait habituellement. Voici le résultat.

Fig.4 Structure4 © DR

II. Balises structurantes

Les balises structurantes ont vu le jour lors du passage du HTML4 au HTML5. Leurs noms ont été choisis d'après l'observation des conventions de nommage les plus utilisées par les webmasters. Ainsi, par exemple, la régularité avec laquelle on retrouve des identifiants nommés « **header** » a convaincu le W3C d'utiliser ce nom de balise.

En outre, d'un point de vue sémantique, il s'agit d'utiliser la balise la plus à même de qualifier l'élément qu'elle comporte. Dès lors la mise en page que l'on a vu précédemment, devient :

```
<div id="wrapper">
    <header>
        <p> header</p>
    </header>
    <nav>
        <p>menu</p>
    </nav>
    <main>
        <p>contenu</p>
    </main>
    <footer>
        <p> footer </p>
    </footer>
</div>
```

Fig. 5 Structure5 © DR

Même si ce n'est pas tout à fait dans notre propos ici, il faut préciser que ces éléments de structures peuvent également être utilisés plusieurs fois dans une même page, mais nous y reviendrons dans un autre cours.

Du côté du CSS on se contente de supprimer l'identifiant, **#header** devient **<header>**, **#footer** devient **<footer>**, etc.

Il nous reste quelques balises à regarder succinctement.

Article : un article est un contenu propre, pouvant être présenté indépendamment de la page et repris par d'autres supports.

Section : une section dans une page ou dans un article, un peu à la manière d'une **div**. Distinction sémantique pour classer des thèmes par exemple.

Aside : un élément d'information permettant d'apporter un éclairage supplémentaire, telle une citation mise en exergue du contenu auquel elle se rapporte.

Notre code d'une page simple pourrait alors devenir.

```
<div id="wrapper">
    <header>
        <p>header</p>
    </header>
    <nav>
        <p>menu</p>
    </nav>
    <main>
        <article>article
            <section>section 1</section>
            <section>section 2</section>
            <section>section 3</section>
        </article>
        <aside>
        </aside>
    </main>
    <footer>
        <p>footer</p>
    </footer>
</div>
```

Fig. 6 Structure6 © DR

Reprenons et ajoutons quelques éléments CSS pour se rendre compte du résultat.

D'abord on doit supprimer les identifiants en trop puisqu'on cible désormais des balises HTML propres. Ensuite, on va reprendre grosso modo le même code que pour les autres balises. On ne va changer que les couleurs de fond.

```css
header{
    min-height:250px;
    background: #F11F33;
    text-align: center;
    border: 1px solid #000;
}
nav {
    min-height:50px;
    background:#00DD11;
    text-align: center;
    border: 1px solid #000;
}

main{
    min-height:450px;
    background: #CD55FF;
    text-align: center;
    border: 1px solid #000;}

article{
    min-height:250px;
    background: #fff;
    text-align: center;
    border: 1px solid #000;
 }
```

Fig.7 Structure7 © DR

```css
section {
    min-height:250px;
    background: #cdcdcd;
    text-align: center;
    border: 1px solid #000;
}

aside{
    min-height:250px;
    background: #CDe;
    text-align: center;
    border: 1px solid #000;
}

footer{
    min-height:150px;
    background: #FDD131;
    text-align: center;
    border: 1px solid #000;
}
```

Fig.8 Structure8 © DR

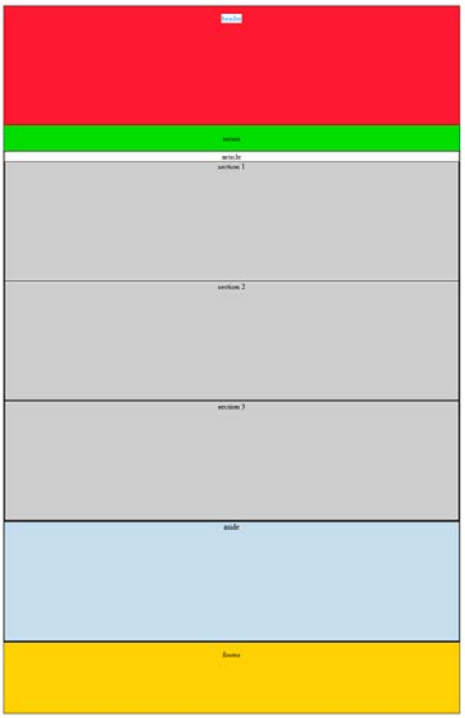

Fig.9 Structure9 © DR

On se rend pourtant compte qu'il est difficile de différencier les sections entre elles, ainsi que l'article. Pour cela, l'usage de **padding** dans la balise **article** va nous permettre de faire la différence entre le parent (article) et les enfants (sections). En outre, en ajoutant une petite marge entre les sections, on va pouvoir visualiser chacune d'elles.

```css
article{
    padding: 20px;
    min-height:250px;
    background: #fff;
    text-align: center;
    border: 1px solid #000;
 }

section {
    margin: 5px;
    padding: 0px;
    min-height:250px;
    background: #cdcdcd;
    text-align: center;
    border: 1px solid #000;
}
```

Fig.10 Structure10 © DR

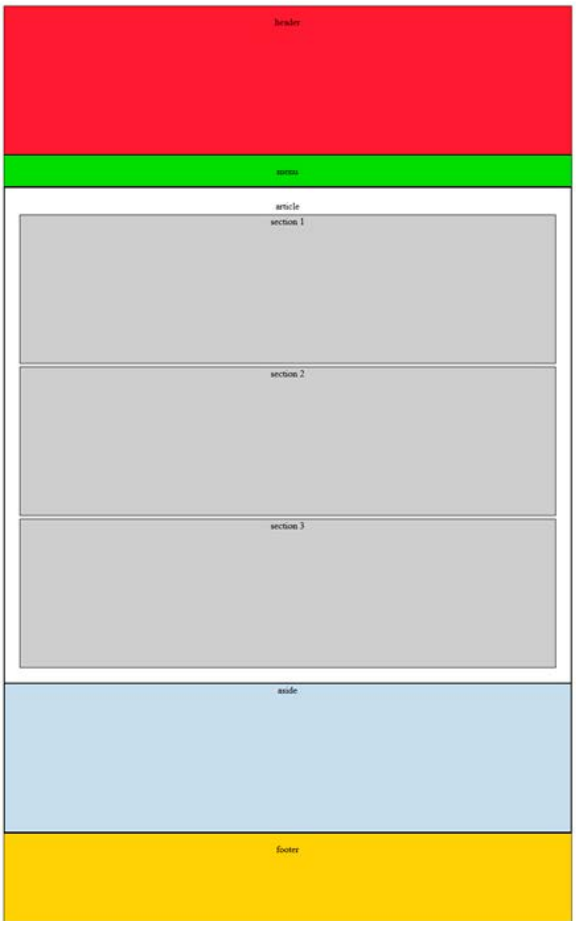

Fig. 11 Structure11 © DR

Vous remarquerez qu'ayant indiqué un **margin** de 5px, c'est-à-dire une marge courant tout autour du bloc, on ne note pas un espace de 10px entre chacun d'eux : l'espace reste de 5px. Ce phénomène est ce que l'on nomme la **fusion des marges**, à savoir que lorsque l'on a deux blocs qui se succèdent l'un au-dessous de l'autre, c'est la plus grande marge des deux qui l'emporte.

III. Conclusion

Ce que l'on remarque à partir d'une telle page, c'est que si l'on peut aérer certaines zones de la page, les possibilités de mise en page sont encore limitées. C'est pourquoi nous devons apprendre à travailler avec des méthodes qui modifient la position des blocs par défaut.