Le modèle des grilles CSS

(ex : page à 3 colonnes, éléments flex)

Rédigé par: Jean-Luc Colson

Date première rédaction: Janvier 2020

SUIVI DES MODIFICATIONS A LA FICHE

Série	Date	Page(s) modifiée(s)	Raison

Contents

Exercice : création d'une page à 3 colonnes contenant des éléments flexibles

Le but de cet exercice est de créer un design de page complexe en réutilisant le maximum de concepts vus jusqu'ici : éléments structurants, flexbox, responsive design, grilles, etc. afin de voir une dernière fois comment les différentes notions vues jusqu'ici peuvent fonctionner ensemble en pratique.

Je vous propose donc de créer à nouveau une page complète qui va suivre deux dispositions différentes selon qu'elle soit affichée sur petits écrans (sur mobile) ou sur un écran de bureau.

Comme nous en avons désormais l'habitude, nous allons commencer par créer la version mobile de la page qui sera la version standard puis utiliserons les Media Queries pour créer une disposition différente pour l'affichage sur bureau.

La page va contenir les éléments suivants :

- 1 en-tête ou header qui va comprendre un menu ;
- 4 blocs de type aside ;
- 1 bloc de contenu de type article qui contiendra lui-même une structure complète ;
- 1pied de page ou footer.

Notre page suivra un design vertical avec une seule colonne pour l'affichage sur mobile et un design sur 3 colonnes pour l'affichage sur bureau.

Structure HTML de notre page

Comme d'habitude, commençons avec la structure HTML de notre page et notamment avec la structure minimale d'une page HTML et sans oublier la meta name="viewport".

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset="utf-8">
        <meta name="viewport"
              content="width=device-width,initial-scale=1.0,user-scalable=no">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>

    </body>
</html>
```

Ensuite, nous allons ajouter un titre principal dans notre page et récupérer la structure du menu créé précédemment qui va très bien nous convenir. Nous allons placer le menu (mais pas le titre) dans un élément structurant header.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset="utf-8">
        <meta name="viewport"
              content="width=device-width,initial-scale=1.0,user-scalable=no">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Une page utilisant les grilles et le flexbox ? Facile !</h1>
        <header>
            <nav>
                <div class="conteneur-nav">
                <label for="mobile">Afficher / Cacher le menu</label>
                <input type="checkbox" id="mobile" role="button">
                  <ul>
                    <li class="deroulant"><a href="#">Cours Complets  </a>
                      <ul class="sous">
                        <li><a href="#">Cours HTML et CSS</a></li>
                        <li><a href="#">Cours JavaScript</a></li>
                        <li><a href="#">Cours PHP et MySQL</a></li>
                      </ul>
                    </li>
                    <li class="deroulant"><a href="#">Articles  </a>
                      <ul class="sous">
                        <li><a href="#">CSS display</a></li>
                        <li><a href="#">CSS position</a></li>
                        <li><a href="#">CSS float</a></li>
                      </ul>
                    </li>
                    <li><a href="#">Contact</a></li>
                    <li><a href="#">A propos</a></li>
                  </ul>
                </div>
            </nav>
        </header>
    </body>
</html>
```

Sous le header, nous allons ajouter nos quatre éléments aside et notre élément article et allons placer tout cela dans un élément section.

Notre élément article va lui-même contenir un header, un menu de navigation interne, un corps qu'on va à nouveau placer dans un élément article et un footer.

```
</header>
<!-- Section-->
<section class="conteneur">
    <aside class="g1"><p>Aside n°1</p></aside>
    <aside class="g2"><p>Aside n°2</p></aside>
    <article class="g3 sousconteneur">
        <header>
            <h1>Titre de l'article</h1>
            <p>Bonne lecture....</p>
            <nav>
                <ul>
                    <li><a href="#">Chapitre 1</a></li>
                    <li><a href="#">Chapitre 2</a></li>
                    <li><a href="#">Chapitre 3</a></li>
                </ul>
            </nav>
        </header>
        <article>
            <p>Apprendre à créer une page avec un design complexe.</p>
        </article>
        <footer>
            <p>A propos de l'auteur: Jean-Luc Colson</p>
        </footer>
    </article>
    <aside class="g4"><p>Aside n°4</p></aside>
    <aside class="g5"><p>Aside n°5</p></aside>
</section>
```

Finalement, nous allons ajouter notre élément footer sous notre élément section.

```
<footer>
    <p>Page HTML/CSS avec grid et flexbox</p>
    <p>&copy; Jean-Luc Colson</p>
</footer>

</body>
</html>
```

Pour cette partie HTML, nous n'allons pas nous attarder sur le contenu comme on a pu le faire dans l'exercice de création d'un CV puisque ce n'est pas ce qui compte ici. Ce qui nous importe ici va être de créer des dispositions différentes. Libre à vous ensuite de placer le contenu que vous voulez dans les différents blocs HTML et de le mettre en forme !

Mise en forme CSS de notre page version mobile

Notre page en version mobile ne va posséder qu'une seule colonne. Les différents éléments vont donc être disposés selon un seul axe. Dans cette situation, on préfèrera utiliser le modèle des boites flexibles plutôt que les grilles.

Commençons déjà par notre traditionnel reset CSS et par ajouter quelques styles globaux.

```css
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
```

Ensuite, nous allons reprendre le code de notre menu créé précédemment et allons faire quelques ajustements.

Tout d'abord, étant donné que notre élément nav est désormais dans un élément header, nous allons plutôt appliquer la position : sticky au header afin que le menu continue à rester collé lorsqu'on défile dans la page.

Nous allons également ici modifier les sélecteurs contenant nav en préférant l'écriture body > header > nav pour bien différencier les styles de notre menu de navigation principal et celui interne à notre élément article.

```css
body > header{
    position: sticky;
    top: 0px;
}
body > header > nav{
    width: 100%;
    height: 42px;
    margin: 0 auto;
}
.conteneur-nav{
    position: absolute;
    width: 100%;
}
body > header > nav input[type=checkbox]{
    display: none;
}
body > header > nav label{
    display: inline-block;
    width: 100%;
    padding: 10px 0px;
    text-align: center;
    background-color: RGBa(220,220,220,0.5);
}
body > header > nav ul{
    display: none;
    list-style-type: none;
    background-color: #555;
}
body > header > nav input[type=checkbox]:checked + ul{
    display: flex;
    flex-flow : column wrap;
}
body > header > nav ul li{
    flex: 1 1 auto;
    text-align: center;
}
body > header > nav > div > ul > li > a{
    color: white;
}
body > header > nav a{
    display: block;
    text-decoration: none;
    color: black;
    padding: 10px 0px;
}
```

Voilà tout pour les ajustements relatifs au menu. Passons maintenant au corps de page qui va utiliser le modèle des boites flexibles.

Ici, nous allons déjà commencer par appliquer un display : flex et un flex-flow : column wrap à notre élément section class="conteneur" pour obtenir une orientation en colonne et transformer ses enfants directs en éléments flexibles.

```
.conteneur{
    display: flex;
    flex-flow: column wrap;
}
```

Ensuite, nous allons simplement nous contenter d'ajouter une couleur de fond et une hauteur minimale à nos différents éléments aside.

```
.g1, .g4{background-color: RGBa(120,205,225,0.4);}
.g2, .g5{background-color: RGBa(225,100,175,0.4);}
.g1, .g2, .g4, .g5{min-height: 100px;}
```

On va maintenant mettre en forme notre élément article class="g3 sousconteneur". Nous allons déjà transformer cet élément en conteneur flexible et allons à nouveau choisir l'axe vertical comme axe principal.

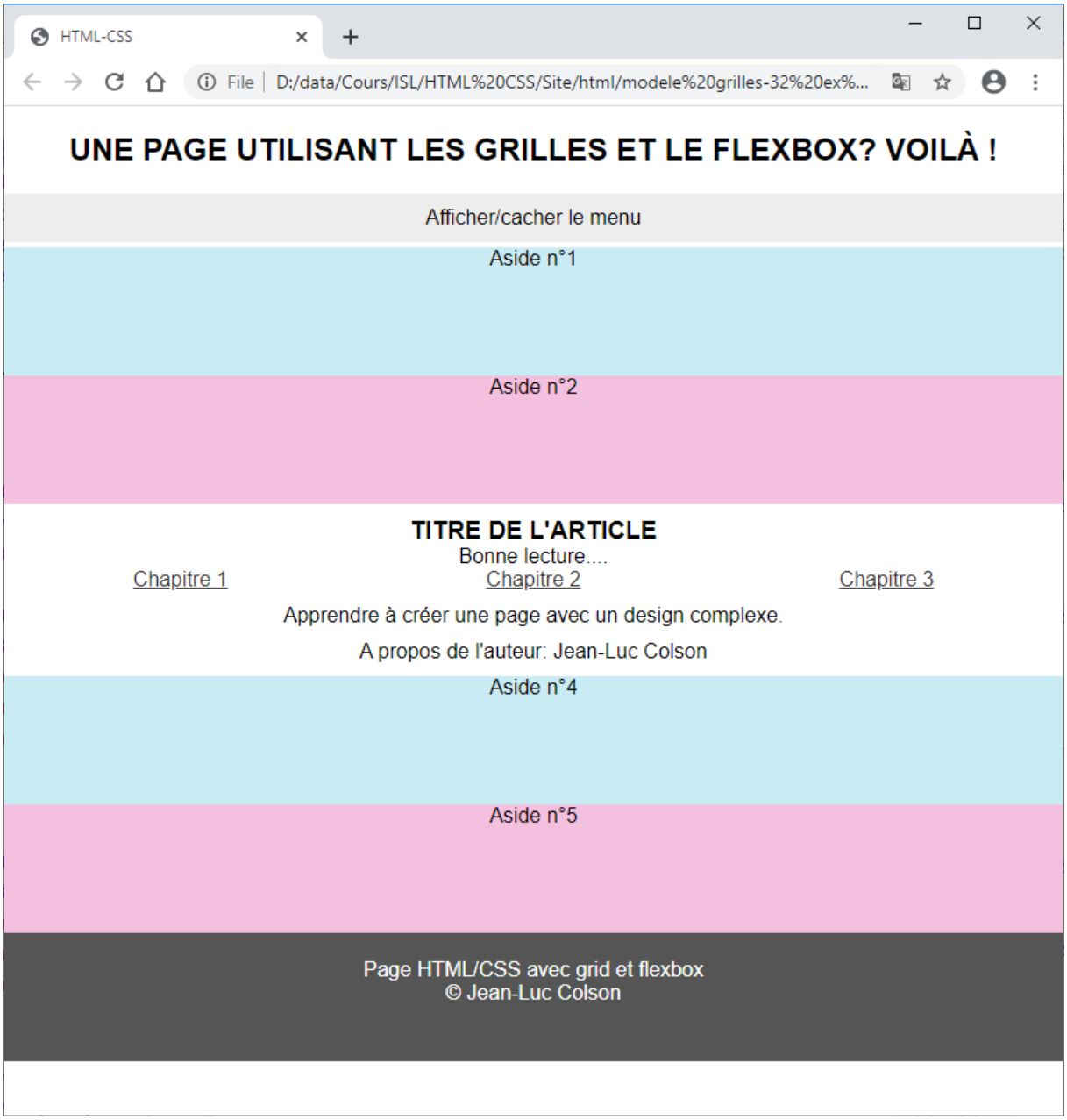
On choisit d'afficher la navigation interne de l'article en ligne. Pour cela, on applique à nouveau un display : flex à notre élément de liste ul qui fait office de navigation et on choisit cette fois-ci l'axe horizontal comme axe principal. On espace régulièrement les éléments dans l'axe principal avec justify-content: space-around.

```css
.sousconteneur{
    display: flex;
    flex-flow: column wrap;
    text-align: center;
    background-color: RGBa(225,215,120,0.4);
}
.sousconteneur h1{
    font-size: 1.2em;
    margin: 10px 0px 0px 0px;
}
.sousconteneur nav ul{
    display: flex;
    flex-flow: row wrap;
    list-style-type: none;
    justify-content: space-around;
}
.sousconteneur nav a{
    display: block;
    color: #333;
}
.sousconteneur article{
    margin: 10px;
}
.sousconteneur footer{
    margin-bottom: 10px;
}
```

Finalement, on applique une mise en forme très basique à notre footer principal qui ne va nous servir ici qu'à afficher une notice de copyright.

```css
body > footer{
    width: 100%;
    height: 100px;
    background-color: #555;
    color: white;
    text-align:center;
    padding: 20px;
}
```

Voilà tout pour la partie mobile. Passons maintenant à la version bureau sur trois colonnes de notre page.



Mise en forme CSS de notre page version bureau

Pour la version bureau de notre page, nous allons vouloir positionner et aligner les éléments sur les deux axes. Nous allons donc préférer l'utilisation des grilles au modèle des boites flexibles pour les blocs de la page.

Nous utiliserons cependant également le flexbox à l'intérieur de certains blocs pour créer des éléments de grille flexibles.

Commençons déjà par définir une règle @media screen and (min-width: 980px) et par remplacer les sélecteurs de notre menu principal nav comme précédemment par body > header > nav.

```css
@media screen and (min-width: 980px){
    .conteneur-nav{
        position: static;
    }
    body > header > nav label, nav input{
        display: none;
    }
    body > header > nav input[type=checkbox]:checked + ul, body > header > nav ul{
        display: flex;
        flex-flow: row wrap;
        background-color: RGBa(220,220,220,0.5);
    }
    body > header > nav ul li{
        position: relative;
    }
    body > header > nav > div > ul > li > a{
        color: black;
    }
    body > header > nav a{
        border-bottom: 2px solid transparent;
    }
    body > header > nav a:hover{
        color: orange;
        border-bottom: 2px solid gold;
    }
    .sous{
        display: none;
        box-shadow: 0px 1px 2px #CCC;
        background-color: white;
        position: absolute;
        width: 100%;
    }
    body > header > nav > div > ul li:hover .sous{
        display: flex;
        flex-flow: column wrap;
    }
```

Ensuite, nous allons cette fois-ci appliquer un display : grid à notre élément section class="conteneur". On va donc ici créer une grille à trois colonnes avec une colonne centrale deux fois plus large que les colonnes gauche et droite. Notre grille va également posséder deux rangées qui devront faire à minima 300px de haut.

```css
    .conteneur{
        display: grid;
        grid-template-columns: 1fr 2fr 1fr;
        grid-template-rows: repeat(2,minmax(300px, 1fr));
    }
```

On va ensuite positionner nos différents éléments dans notre grille. On va ici vouloir que l'élément article occupe toute la colonne centrale et allons positionner les éléments aside de chaque côté.

```css
.g1{
    grid-column: 1 / 2;
}
.g2{
    grid-column: 1 / 2;
    grid-row: 2 / 3;
}
.g3{
    grid-column: 2 / 3;
    grid-row: 1 / 3;
}
.g4{
    grid-column : 3 / 4;
}
.g5{
    grid-column : 3 / 4;
    grid-rows: 2 / 3;
}
```

Et voilà tout ! Notre élément article est toujours un conteneur flexible et les propriétés définies pour l'affichage mobile s'appliquent toujours ici et nous conviennent très bien pour l'affichage sur bureau. Pas besoin d'aller plus loin donc.

Nous avons donc créé une page totalement responsive et avec une structure d'affichage complexe en utilisant certaines des notions les plus récentes du HTML et du CSS !