PARTIE 2 : mise en page d'une page HTML

Objectifs :

- Présentation du langage CSS
- Principe de séparation du contenu et de la mise en forme
- Analyser et modifier la mise en forme d'une page web grâce au langage CSS

Introduction

Jadis, dans des temps reculés où l'internet et les informations qu'il contenait n'intéressaient pas un large public, l'esthétisme des pages était très limité et on pouvait se contenter d'une mise en page sommaire.
Depuis, le web s'est démocratisé et le rendu des pages est devenu plus complexe, plus travaillé. Les balises du langage HTML ont évolué et se sont multipliées… multipliées…multipliées…(à partir de HTML 3)
A un tel point que la lecture des fichiers et leur maintenance (obligatoire pour toute mise à jour) est devenu un véritable cauchemar. Les temps de chargement s'en trouve également fortement impactés (on n'avait pas de fibre, ni d'ADSL, mais une simple ligne 56kb…).

Pour cela, le W3C (World Wide Web Consortium) a mis en place une technique séparant le fond et la forme d'une page web : laissant le fond au langage HTML, ils ont adopté **les feuilles de style en cascade (Cascading Style Sheets)** pour la forme.
Une feuille de style pouvant être reliée à plusieurs page html.

Déclaration d'une feuille de style

Il y a 2 façons d'introduire une feuille de style auprès du ficher HTML.

- Soit directement dans le fichier HTML lui-même grâce à une balise <style>…</style> placée dans le *head* du fichier. Exemple :

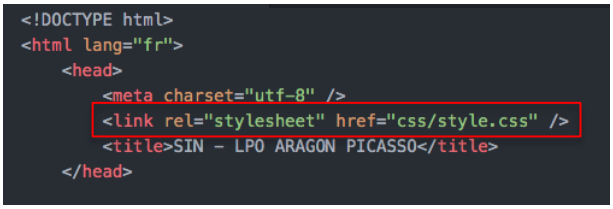
```
<!DOCTYPE html>
<html>
<head>
<style>
p {
    color: blue;
    text-align: center;
}
</style>
</head>
<body>
<p>Bonjour les gens</p>
<p>Ce texte est mise en forme grâce à du CSS</p>
</body>
</html>
```

instructions de mise en forme

Bonjour les gens

Ce texte est mise en forme grâce à du CSS

- Soit en créant un lien entre la page HTML et un fichier .css contenant la mise en forme. L'intérêt de cette méthode est de pouvoir lier plusieurs pages HTML à un unique fichier de mise en forme. En cas de modification du CSS, toutes les pages du site sont impactées par la mise en forme. Cette méthode est à préconiser.



Syntaxe de base : modification de la mise en forme des éléments de la page.

Le principe de base d'un CSS est de déclarer les propriétés d'un sélecteur. Toute la nuance est dans le choix du sélecteur.
La syntaxe de base est toujours la même :

```
h1                          nom de l'élément du HTML visé par la propriété
{
color : black ;             propriété et valeur séparé par « : » et terminé par « ; »
font-size : 12px ;
}
```

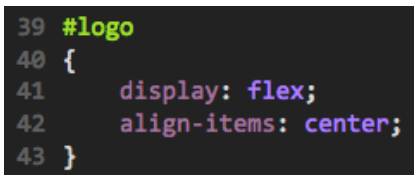
Les propriétés sont nombreuses.

Elles permettent de modifier une police d'écriture, une taille de police, une marge, …. Vous pouvez retrouver toutes les propriétés sur cette page :

https://www.w3schools.com/cssref/css3_pr_align-content.asp (sélectionner la propriété dans le menu de gauche)

Les sélecteurs

Permettent de choisir sur quel(s) élément(s) s'applique la propriété. Et là et la puissance du CSS…. A ce stade, on peut choisir de viser :

Sélecteur	Elément type visé	exemple
Une balise de la page : on note le nom de l'élément visé.	Header ou body ou footer ou article ou ….	
Un élément unique : on note le nom du **id** en le faisant précéder par **#.**	Id= ''logo ''	
Un type d'élément : on définit dans ce cas une **classe** dans le fichier HTML sous la forme **class= ''nom ''**	class=''liens '' dans ce cas, on définit la couleur par défaut de tous les éléments de la classe **liens** de la page et également la couleur dans le cas où l'élément est survolé (hover).	

```
3  body
4  {
5    background: grey;
6    font-family: "Arial", sans-serif;
7    color: #181818;
8  }
```

```
39 #logo
40 {
41      display: flex;
42      align-items: center;
43 }
```

```
158 .liens
159 {
160      color: #fcfbfb;
161 }
162
163 .liens:hover
164 {
165    color: #dc3131;
166 }
```

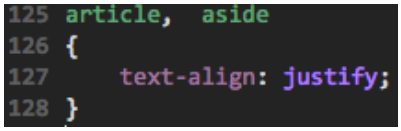
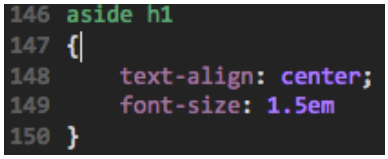
- **Application** : modifier les valeurs des propriétés afin de définir les couleurs de la page et la rapprocher du résultat final proposé en dernière page. Pour les couleurs, on pourra s'aider de cette page :

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Colors/Color_picker_tool

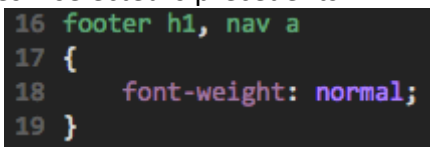
Remarque : Le CSS utilise un codage hexadécimal pour représenter les couleurs (bien que l'on puisse utiliser en clair les couleurs de base, red, white, grey, yellow,…).

- **Question** : sur combien de bits est codée la couleur en CSS ?

On peut également affiner la sélection ou réaliser des sélections multiples :

Sélecteur	Élément type visé	exemple
Plusieurs éléments de la page pour une même propriété	Header ou body ou footer ou article ou …. **Séparés par une virgule**	
Un élément "enfant" d'un autre. La propriété ne s'appliquera à ce sélecteur uniquement s'il est contenu dans cet élément parent.	Ne vise que les éléments **h1** qui sont contenu dans un **aside.** Pas les autres…	

On peut également imbriquer les 2 sélecteurs précédents :



Vise les éléments h1 du footer ET les élements a (les liens) du nav.

Le CSS, comme outil de mise en page

Jusqu'ici, nous avons vu que la feuille de style permet de mettre en forme la page web en modifiant la taille du texte, sa couleur, ….

Le langage CSS permet également de réaliser l'organisation de votre page.
Il existe différentes façons d'organiser la page mais ici, nous allons étudier 2 solutions :

- le « grid-layout »
- le « flexbox »

Remarque : On pourrait utiliser l'un sans l'autre mais on va voir que l'on peut utiliser les 2 conjointement afin de faciliter la mise en page. Après c'est une histoire de goût.

Le grid Layout (positionnement en grille)

Le concept de grille est de diviser l'espace de la page en zone, comme on le fait dans un tableau en définissant des lignes et des colonnes.

Pour cela, on définit un élément conteneur organisé sous forme de grille avec la déclaration ***display : grid*** puis on définira l'organisation de cette grille, en spécifiant la place occupée par chaque élément du fichier HTML (balise, id ou classe).

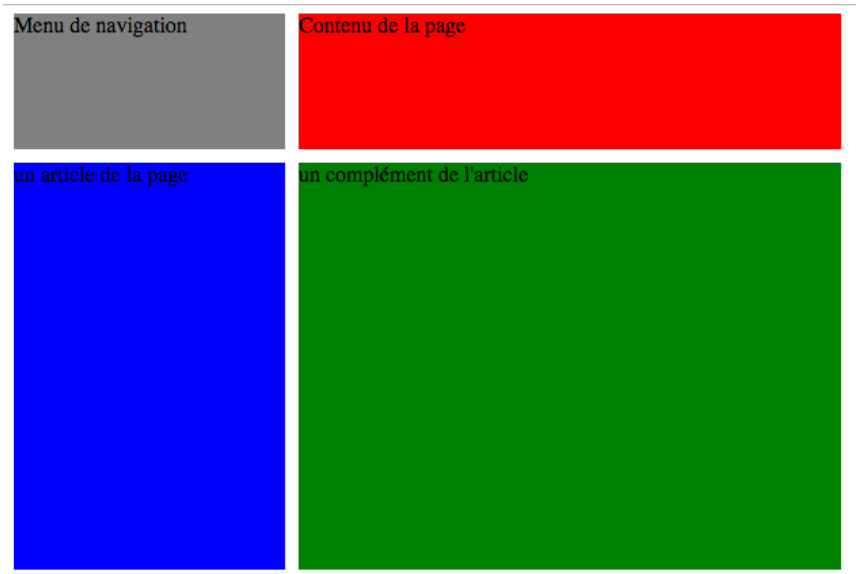
Exemple :

HTML

```
<body>
  <nav>Menu de navigation</nav>
  <section>Contenu de la page</section>
  <article>un article de la page</article>
  <aside>un complément de l'article</aside>
</body>
```

CSS

```
body {
  display: grid;
  grid-template-columns: 200px 400px;
  grid-template-rows: 100px 300px;
  gap: 10px;
}
nav {
  grid-column: 1;
  grid-row: 1;
  background-color: grey;
}
section {
  grid-column: 2;
  grid-row: 1;
  background-color: red;  ;
}
article {
  grid-column: 1;
  grid-row: 2;
  background-color: blue;
}
aside {
  grid-column: 2;
  grid-row: 2;
  background-color: green;
}
```



```css
body {
  display: grid;
  grid-template-columns: 200px 400px;
  grid-template-rows: 100px 300px;
  gap: 10px;
}
nav {
  grid-column: 1;
  grid-row: 1;
  background-color: grey;
}
section {
  grid-column: 2;
  grid-row: 1;
  background-color: red;  ;
}
article {
  grid-column: 1;
  grid-row: 2;
  background-color: blue;
}
aside {
  grid-column: 2;
  grid-row: 2;
  background-color: green;
}
```

Body est conteneur
- On le déclare en grille
- Composée de 2 colonnes de largeur 200px et 400 px
- Et de 2 lignes de hauteur 100px et 300px
- Espacées d'un espace (gap = trou) de 10px

L'élément *nav*
- Est situé dans la 1ère colonne
- Et dans la 1ère ligne
- Son fond est de couleur grise

L'élément *section*
- Est situé dans la 2ème colonne
- Et dans la 1ère ligne
- Son fond est de couleur rouge

L'élément *article*
- Est situé dans la 1ère colonne
- Et dans la 2ème ligne
- Son fond est bleu

L'élément *aside*
- Est situé dans la 2ème colonne
- Et dans le 2ème ligne
- Son fond est vert.

Remarque : les dimensions des lignes et colonnes peuvent se déclarer en utilisant plusieurs arguments (px=pixel, %=%, fr=fraction de l'espace restant, …)
Autre solution : on peut placer les éléments de la page en utilisant des zones cible

```css
body {
  display: grid;
  grid-template-columns: 200px auto;
  grid-template-rows : 100px 300px;
  grid-template-areas:  "n  s"
                        "ar as";
  gap: 10px;
}
nav {
  grid-area: n;
  background-color: grey;
}
section {
  grid-area: s;
  background-color: red;  ;
}
article {
  grid-area: ar;
  background-color: blue;
}
aside {
  grid-area: as;
  background-color: green;
}
```

Body est conteneur
- On le déclare en grille
- Composée de 2 colonnes de 200px et le reste de la page
- Et de 2 lignes de hauteur 100px et 300px
- Et on partage l'espace de la page en zones « areas »
- Séparée par des espaces de 10px.

L'élément *nav*
- Est situé dans la zone « n »
- Son fond est de couleur grise

L'élément *section*
- Est situé dans zone « s »
- Son fond est de couleur rouge

L'élément *article*
- Est situé dans la zone « ar »
- Son fond est bleu

L'élément *aside*
- Est situé dans la zone « as »
- Son fond est vert.

Pour modifier la place des éléments, il suffit de changer leur zone cible (« ar » => « as »).
On peut insérer une colonne ou une ligne vide grâce au caractère « . ».

Pour mieux comprendre

Je vous propose de vous rendre sur cette page et de participer à un petit jeu….

Aide : https://github.com/billfienberg/grid-garden

Le Flexbox Layout Module (modèle de boîte flexible)

Préambule :
Comme le grid, ce module CSS permet de réaliser une organisation complète de votre page web. Cependant, je trouve que son utilisation pour l'organisation dans le sens vertical (lignes) est moins facile que le grid. Pour cela, je m'en sers essentiellement pour l'organisation horizontale de mes éléments grid. Mais c'est personnel….

Principe de fonctionnement

Philosophiquement, il est presque identique à la technologique Grid : on définit un conteneur et on organise les éléments de ce conteneur.

Pour bien appréhender son fonctionnement, je vous propose un petit jeu :

Aide : https://www.youtube.com/watch?v=fMulPRHsNlk

Application

- Définir un conteneur ***grille1*** dans le CSS répondant aux exigences suivantes :

- 3 colonnes de largeur égale (unité fr conseillée)
- un gap de **1em** entre les colonnes
- un schéma de zone du type :

header	header	header
banniere	banniere	banniere
article	article	aside
footer	footer	footer

- une organisation **flex** pour les éléments : **header, footer, nav** (direction row pour ce dernier) et l'id **social** et l'id **listes_sites.**

On peut remarquer que la présentation de la page « page1.html » a été modifié en même temps que vous avez modifié la feuille de style

- Proposer une solution permettant de définir une présentation différente pour la page « page1.html » et la page « index.html » en utilisant le grid layout.

Aperçu de la page web finale