5 minutes

Mise en place du header

Modification de `index.html`

Nous allons commencer par mettre en place la structure de la partie `header` de notre page :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Café Florette | Accueil</title>
    <link rel="stylesheet" href="css/styles.css" />
    <link
      href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,700&d
      rel="stylesheet"
    />
  </head>
  <body>
    <header>
      <nav>
        <h1><a href="./index.html">CAFE FLORETTE</a></h1>
        <ul>
          <li><a href="./index.html">Accueil</a></li>
          <li><a href="./where.html">Nous trouver</a></li>
          <li><a href="./contact.html">Contact</a></li>
        </ul>
      </nav>
    </header>
  </body>
</html>
```

Nous ajoutons un lien pour charger la police `Open Sans` depuis une `API` de `Google`. Nous avons sélectionné trois graisses : 300, 400 et 700.

Dans la partie `body` nous créons un élément sémantique `header` dans lequel nous plaçons un élément sémantique `nav` car il contiendra les liens pour naviguer sur les pages de notre site.

Dans cet élément `nav`, nous plaçons le titre de la page `h1` qui est cliquable car nous y imbriquons un élément `a`.

Ainsi, les utilisateurs seront redirigés sur la page d'accueil lorsqu'ils cliquerons sur le titre.

Ensuite, nous créons une liste non ordonnée `ul`, dans laquelle nous plaçons trois `list items`. Chaque `li` contient un lien dans un élément `a` pour pouvoir naviguer vers les autres pages du site.

Raccourci `Emmet`

Nous allons voir un nouveau raccourci `Emmet` qui permet de grouper des éléments pour appliquer par exemple `*`.

Ici nous voulons créer trois fois des `li` dans lesquels nous avons des `a` :

Modification de `styles.css`

Nous allons ajouter quelques règles `CSS` dans notre fichier `styles.css` :

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

html,
body {
  font-family: "Open Sans", sans-serif;
}

h1,
h2,
h3 {
  margin-bottom: 20px;
}

a {
  color: #222;
  text-decoration: none;
}
```

Nous commençons par utiliser la police `Open Sans` chargée depuis Google pour tous nos éléments. Nous définissons une famille de polices de repli dans le cas où la police ne pourrait être chargée.

Ensuite, nous remplaçons les règles `CSS` par défaut pour les éléments `a` : à savoir le soulignage et le bleu.

Nous enlevons le soulignage en faisant `text-decoration: none;` et nous changeons la couleur avec la propriété `color`.

Modification de `index.html`

Nous allons maintenant attaquer le style de notre `header` :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Café Florette | Accueil</title>
    <link rel="stylesheet" href="css/styles.css" />
    <link
      href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,700&d
      rel="stylesheet"
    />
  </head>
  <body>
    <header class="header">
      <nav>
        <h1><a href="./index.html">CAFE FLORETTE</a></h1>
        <ul>
          <li><a class="active" href="./index.html">Accueil</a></li>
          <li><a href="./where.html">Nous trouver</a></li>
          <li><a href="./contact.html">Contact</a></li>
        </ul>
      </nav>
    </header>
  </body>
</html>
```

Pour cela, nous ajoutons deux classes qui nous permettront de définir des règles `CSS`.

Une première classe `header` qui nous permettra spécifiquement de cibler ce `header` (pour rappel, nous pouvons avoir plusieurs éléments sémantiques `header` dans une même page).

Une seconde classe, `active`, qui nous permettra de définir un style particulier pour le lien de l'accueil. Cela permettra d'indiquer à l'utilisateur qu'il a sélectionné le lien "Accueil" et qu'il est sur la

page correspondante.

Modification de `styles.css`

Nous allons maintenant passer aux règles de style :

```
* {
  margin: 0;
}

html,
body {
  font-family: "Open Sans", sans-serif;
}

h1,
h2,
h3 {
  margin-bottom: 20px;
}

a {
  color: #222;
  text-decoration: none;
}

/* Header */
.header {
  background: black;
  overflow: auto;
  line-height: 1.7em;
}

.header nav h1 {
  float: left;
  padding: 20px;
  margin-bottom: 0;
}

.header nav h1 a {
  color: white;
}

.header nav ul {
  float: right;
```

```
  list-style: none;
}

.header nav ul li {
  float: left;
}

.header nav ul li a {
  color: white;
  padding: 20px;
  display: block;
}

.header nav ul li a:hover,
.active {
  background: #444;
}
```

Nous allons reprendre ensemble les éléments essentiels.

Pourquoi avons-nous dû utiliser `overflow-auto` sur le parent des éléments positionnés en `float` ?

Le `container header` ne contient que des éléments qui sont positionnés en `float`.

Or, pour rappel, lorsque nous plaçons un élément en `float`, **il est retiré du flux normal des éléments**.

Du coup, l'élément `header` fait comme si il n'y avait pas d'éléments et il n'a donc pas de hauteur (pour rappel les éléments de bloc prennent la hauteur de leur contenu).

L'utilisation de `overflow: auto` permet la création d'un contexte de formatage de blocs et oblige l'élément à prendre en compte les éléments positionnés en `float`.

Ne vous attardez pas trop sur ce point car ce n'est pas l'approche la plus moderne. Nous verrons comment il est possible de réaliser la même chose avec le `CSS` le plus moderne lorsque nous verrons les boites flexibles et les grilles dans les chapitres suivants.

Sachez que c'était la meilleure manière de faire sans ces deux modules `CSS` plus modernes. Il est encore possible que vous la rencontriez sur des sites développés il y a quelques temps.

Pourquoi devons nous définir une `line-height` ?

Par défaut, les navigateurs prennent `1.2` (ce qui signifie que la valeur de `font-size` de l'élément est multiplié par 1.2).

Donc comme un `h1` dans une `nav` a une taille de `1.5em` par défaut sa `line-height` sera de `1.2*1.5*16px` donc `28.8 px`. Car pour rappel, dans un navigateur, `1em` vaut `16px` par défaut.

Et comme les éléments ont une taille de `1 em` soit `16px`, les liens auront une `line-height` de `1.2*1*16 px` soit `19.2 px`.

Pour donner à nos éléments la même `line-height`, nous la fixons donc à `1.7em` ce qui permet que les clics sur les liens puissent se faire sur toute la hauteur du `header`.

Nous ne détaillerons pas plus le comportement des éléments en `float` car ce n'est pas utile pour vous. Nous verrons très en détails les grilles et les boites flexibles à la place.

Etape 2 du projet

Retrouvez le code du projet à cette étape :
https://codesandbox.io/embed/htmlcss-c4-l2-ubn5l