Table des matières

Chapter 1

HTML

1.1 Choix des outils

1.1.1 Production des fichiers

Nous allons produire notre site en local, c'est-à-dire sur un ordinateur du lycée.

Il existe plusieurs solutions pour créer un fichier HTML, y compris l'export dans des logiciels de traitement de texte.

La solution retenue ici est l'utilisation d'un éditeur de texte pour plusieurs raisons:

- La plupart des logiciels de traitement de texte ou des éditeurs spécialisés produisent un code difficilement lisible
- Ils ne séparent pas toujours HTML et CSS et rendent donc plus compliquée l'harmonisation des pages, ainsi que la modification de l'apparence globale du site.
- Ces éditeurs font des erreurs de logique, qui rendent parfois le code non valide.
- Les éditeurs ont des limitations diverses.
- Notre objectif est aussi de comprendre ce qui se cache derrière, et pas seulement de produire un site.
- Pour optimiser le référencement, une maitrise du HTML est indispensable.

Le choix de l'éditeur de texte est personnel, et vous pouvez même vous contenter du bloc-notes si vous êtes sous Windows.

Néanmoins, un éditeur qui gère la coloration syntaxique est plus pratique.

Vous pouvez essayer notepad++ (Windows), Geany (multiplateforme), brackets.io (multiplateforme), atom.io (multiplateforme)...

1.1.2 Vérification du code

Il est indispensable de vérifier que le code tapé correspond bien à ce qui était souhaité.

Idéalement, le code que vous avez tapé est valide, et les navigateurs le transcriront parfaitement.

En pratique, même si votre code est validé, certains navigateurs ne le prendront pas en charge comme attendu.

La vérification doit donc être double:

- Voter code doit être validé:

  Il existe pour cela des outils en ligne, comme `http://validator.w3.org/`

- pour vérifier que le résultat est conforme à vos souhaits, il faut tester, en pratique, sur différents navigateurs, y compris sur smartphone et tablette.

1.2 Premières balises

Pour notre premier exemple, nous allons utiliser une page fictive, simplifiée par rapport aux pages réellement présentes sur le Web, et qui respecte les normes.

Pour progresser, vous êtes encouragés à regarder, et modifier, le code de pages existantes.

Nous verrons la démarche après notre exemple. La première étape est d'ouvrir votre éditeur de texte préféré (ou celui qui est installé sur votre PC). Créez un nouveau fichier et enregistrez le sous le nom page1.html Ce fichier pourra être supprimé par la suite.

Dans un objectif de production, utilisez si possible des noms explicites, et évitez les fichiers nommés test.
Evitez aussi les caractères spéciaux, les espaces, les accents...

Très important: la page d'accueil de votre site va s'appeler index.html

Même si la plupart du temps, les navigateurs sont assez permissifs, n'oubliez pas que le HTML est un langage de balises. Ces balises permettent de structure le document.

Il existe de nombreuses balises. La plupart d'entre elles vont encadrer des zones du document. La balise située avant est considérée comme une balise ouvrante. Celle située après est une balise fermante. Par exemple, pour indiquer que le contenu du fichier est en HTML, il faut utiliser une balise: la balise <html> pour ouvrir et la balise </html> pour fermer. La différence entre une balise ouvrante et une balise fermante est le /

Certaines balises vont indiquer que le contenu est un texte, une image. Mais il existe aussi des balises pour définir la structure même du fichier.

1.2.1 Un exemple

On considère le code suivant.

Faites un copier-coller dans votre fichier texte, sauvegardez-le et afficher le fichier dans votre navigateur.

```
<!-- Mon premier fichier HTML -->
<!DOCTYPE html>
<html lang="fr">
 <head>
  <meta charset="utf-8" />
  <title>Une page un peu simplifiée</title>
 </head>

 <body>
  <!-- Contenu -->
```

```
    <h1>Mon titre Principal</h1>

    <p>Bienvenue sur ma page simplifiée.</p>
    <p> Si vous         cliquez sur les liens, vous n'irez nulle part.<br>
    Juste un petit passage à la ligne</p>

    <p><img src="http://www.lyc-jean-michel.ac-besancon.fr/sites/www.lyc-jean-michel/IMG/arton101.
          alt="logo du lycée" width="40"></p>

   <!-- menu -->
    <ul class="navbar">
     <li><a href="#">Accueil</a></li>
     <li><a href="liens.html">Mes Liens</a></li>
    </ul>

  </body>
</html>
```

Modifiez les différentes informations de ce fichier.
A quoi correspondent-elles?
Quel est l'impact de vos modifications?

1.2.2 Décryptage de l'exemple

Comme vous avez pu le constater, tout le texte n'est pas affiché.

Les informations sont gérées différemment selon le type de balises utilisé.

Examinons l'allure générale du code: Vous pouvez remarquer que certaines balises ont été décalées vers la droite, ce qui n'a eu aucun effet sur la présentation de la page.

En effet, ce choix a été fait pour rendre le code plus lisible, mais le nombre d'espaces ou de retour à la ligne n'est pas important en HTML. Cette démarche est donc conseillée, mais pas obligatoire.

Pour vous simplifier le travail, n'hésitez pas à présenter les informations approximativement comme vous souhaitez qu'elles le soient. Cela n'aura aucun effet pratique sur le navigateur, mais vous aidera à vous repérer.

Examinons maintenant le code ligne par ligne:

1.2.2.1 La première ligne est un commentaire

Il s'agit d'un texte qui n'est pas destiné à être vu par l'utilisateur final, ni à être interprété par le navigateur. Il sert d'aide-mémoire au développeur.

Même s'ils ne sont pas obligatoires, les commentaires sont conseillés dès qu'on doit faire du codage.
En HTML, les commentaires s'écrivent sous la forme:

```
<!-- Ceci est mon commentaire
Sur plusieurs lignes -->
```

Attention, la source d'une page web est visible.

Le commentaire n'apparaitra pas à l'affichage de la page, mais n'importe qui peut le voir en affichant le code source.

1.2.2.2 doctype

Il s'agit d'un cas un peu particulier. Son rôle est d'indiquer qu'il s'agit d'une page HTML. C'est une balise qui n'a pas de balise fermante.

1.2.2.3 balises <html> et </html>

Elles englobent le contenu de toute la page.

On précise, à l'aide de lang="fr", que la page est en français.

Ce n'est pas obligatoire, mais recommandé, car utilisé par les moteurs de recherche.

1.2.2.4 balises <head> et </head>

Ce sont les balises qui définissent l'en-tête du document. Ces informations ne seront pas visibles dans la page, mais permettent de donner des précisions au navigateur: encodage, titre, mots-clés...

On observe que les balises peuvent être imbriquées les unes dans les autres: meta et title sont à l'intérieur de la zone définie par head.

Attention à l'ordre de fermeture: des balises peuvent être imbriquées les unes dans les autres, mais la dernière ouverte est la première fermée.

```
<html>
 <head>
 </html>
</head>
```

Cette structure n'est pas valide, car html n'encadre pas <head>

1.2.2.5 balise <meta>

Cette balise est destinée à donner des informations particulières sur la page, qui ne seront pas affichées.

Elle est un peu différente des balises vues jusqu'ici pour 2 raisons:

- elle n'a pas de balise fermante. On parle de balise auto-fermante. Pour préciser qu'il s'agit d'une balise auto-fermante, on écrit / à la fin de la balise.
- entre < et >, il n'y a pas que le nom de la balise.

  On dit que la balise possède un attribut. Ces attributs peuvent être de différents types. Leur valeur est entre guillemets.

  Ici, il s'agit de préciser l'encodage du fichier: <meta charset="utf-8" />.

  Si cette balise n'est pas présente, le navigateur n'utilisera pas toujours le bon encodage.

  La conséquence est la présence de caractères étranges en lieu et place des caractères accentués, par exemple.

  Attention: Nous utiliserons l'encodage utf-8. Ce n'est pas forcément l'encodage par défaut dans votre éditeur de texte.

  Si ce n'est pas le cas, n'hésitez pas à le changer. L'encodage utf-8 permet d'éviter pas mal d'ennuis.

1.2.2.6 balises <title> et </title>

Elles permettent de définir le titre de la page.

Il n'est pas affiché dans la page, mais dans la barre de titres, ainsi que dans l'affichage des résultats d'un moteur de recherche.

Cette balise est donc très importante. Ne l'oubliez pas!

1.2.2.7 balises <body> et </body>

L'en-tête étant terminée, il est temps de passer au corps du document, c'est-à-dire ce qui sera affiché.

1.2.2.8 balises <h1> et </h1> : des balises de titre

Il ne s'agit pas ici du titre de la page, mais de titres affichés dans le texte. Là encore, les moteurs de recherche utilisent ces informations pour détecter les informations importantes sur votre page. Même si pratiquement, les titres sont plus gros que les textes "normaux", ne vous servez pas de titre pour augmenter la taille de la police de caractères. Ce sera le rôle du CSS.

Il existe 6 niveaux de titre, allant de h1 (le plus important) à h6 (le moins important)

```
<h1>Mon titre principal </h1>
 <h2>Mon premier sous-titre</h2>
 <h2>Mon second sous-titre</h2>
```

1.2.2.9 balises et : des balises de paragraphe

Chaque paragraphe est délimité par les balises et .

Si vous effectuez des retours à la ligne dans votre fichier texte, ils seront ignorés par le navigateur, tout comme les espaces multiples.

Profitez-en pour rendre votre fichier plus clair en utilisant l'indentation

```
<p>Bonjour.     Comment

allez-vous?</p>
```

sera simplement affiché: Bonjour. Comment allez-vous?

Par contre,

```
<p>Bonjour.</p>
<p>Comment allez-vous?</p>
```

sera bien affiché:

```
Bonjour.
Comment allez-vous?
```

Si vous avez besoin d'effectuer un saut à la ligne en restant dans le même paragraphe, vous pouvez utiliser la balise <br>:

```
<p>Bonjour.<br>
Comment allez-vous?</p>
```

Le texte sera bien affiché sur 2 lignes, mais considéré comme le même paragraphe.

1.2.2.10 balises <img>: des balises images

Il est évidemment possible d'insérer des images dans une page web.

On utilise pour cela la balise <img>.

Cette image est en général placée dans une balise p (complément: ou une autre balise de type block).

Cette balise prend en général au moins 2 attributs:

- src L'attribut src prend pour valeur l'adresse de l'image (absolue ou relative)
- alt L'attribut alt prend pour valeur un texte qui sera affiché si l'image ne peut être affichée, ou pour les non-voyants

En pratique,

```
<img src="montage.jpg" alt="C'est la description de mon image"/>
```

Les navigateurs supportent différents formats.

Le choix d'un format dépend du type d'images et de son utilisation.

En général, on utilise le format JPG, ou PNG pour des images qui doivent gérer la transparence.

Evitez les images trop lourdes, sauf si votre site est spécialisé dans les photos haute qualité.

Si nécessaire, il est possible de changer la taille d'affichage de l'image en ajoutant un attribut width ou height, comme dans notre exemple.

Cette solution est acceptée, mais il est conseillé de faire ces adaptations dans le fichier CSS.

1.2.2.11 balise

Cette balise permet de définir une liste non ordonnée (unordered list).

La liste se termine par

Chaque élément de la liste est délimité par des balises <li> et

On définit de même une liste ordonnée avec des balises <ol> et .

1.2.2.12 balise <a>

Cette balise est à la base du nom HTML (H pour hypertext).

Elle permet d'établir un lien vers une autre page web.

Sa structure est la suivante:

```
<a href="adresse du lien">Texte à afficher ou image</a>
```

Suivant la situation, l'adresse peut être absolue ou relative:

Une adresse est dite absolue quand on donne l'adresse complète. Elle est dite relative quand on donne sa position par rapport à l'adresse actuelle.

Une analogie simple est de voir ce qui se passe pour de vraies adresses: Supposons que vous souhaitiez envoyer un courrier (papier) à quelqu'un qui vit loin de vous.

Sur l'enveloppe, vous allez indiquer:
Nom du destinataire
Numéro de rue, rue
Code postal, Ville
Pays

Il s'agit d'une adresse absolue.

Par contre, si vous êtes avec des amis et que vous croisez votre voisin.
Si on vous demande de qui il s'agit, vous n'allez pas donner son nom et son adresse complète.
Vous allez indiquer, pour être précis: "c'est mon voisin (de droite)".

Il s'agit d'une adresse relative: vos amis, qui savent où vous habitez, savent maintenant comment aller chez cet individu.

Pour un site web, c'est la même chose:

- si vous pointez un lien vers une page de votre site, le fichier est dans le même dossier, il n'est donc pas souhaitable de donner l'adresse complète (qui pourrait changer si vous déménagez votre site). Vous allez donc utiliser une adresse relative
- si vous pointez un lien sur un autre site web, vous choisirez une adresse absolue

Exemples:

```
<!-- lien absolu vers la page d'accueil du site du lycée -->
<a href="http://www.lyc-jean-michel.ac-besancon.fr/">page d'accueil</a>

<!-- lien relatif vers la page page2.html du même dossier -->
<a href="page2.html">page dans le même dossier</a>

<!-- lien relatif vers la page page3.html dans un sous-dossier nommé sous-->
<a href="sous/page3.html">page dans un sous-dossier</a>

<!-- lien relatif vers la page page4.html d'un dossier parent -->
<a href="../page4.html">page dans un dossier parent</a>
```

1.2.3 Balises div et span

Les balises div et span servent à créer des conteneurs pour des zones d'une page qui doivent être regroupées, sans pour autant correspondre à une catégorie précédente.

On pourra donc appliquer à toute une zone de la page le même style.

Pour cela, il faudra donner un identifiant à la zone, à l'aide d'un attribut id ou d'un attribut class - voir bases CSS

Leur rôle est donc similaire.

La différence entre les deux est la nature de la zone.

 s'applique à des éléments de type bloc (block), alors que span s'applique à des éléments de type en ligne (inline).

Quelle est cette distinction ?:

- un élément de type inline ne démarre pas une ligne et ne prend que la place nécessaire.

  Une balise de type <a> correspond à un élément de type inline: on applique cette balise à un ou plusieurs mots seulement.

- un élément de type block occupe des lignes entières (dans son élément parent).

  Une balise de type <p> correspond à un élément <block>: entraine un passage à la ligne et occupe toute la place disponible en ligne

  Application

```
<!-- exemple d'utilisation de balises div et span -->
<!-- Zone 1 -->
<div id="mazone1">
<p>Mon premier paragraphe</p>
<p>Mon second paragraphe</p>
</div>


<!-- Zone 2 -->
<div id="Mazone2">
<p>Mon troisième paragraphe</p>
<p>Le quatrième paragraphe contient un <span id="motcle1">mot</span>
qui a un rôle différent.</p>
</div>

<p>Mon cinquième paragraphe <span id="motcle2">aussi.</span></p>
```

On observe que dans une balise div, il peut y avoir un ou plusieurs paragraphes, et même des balises span.

Par contre, les balises span peuvent aussi être à l'extérieur, sans balise div

Pour créer une ancre, on n'utilise pas une balise, mais un attribut, nommé id, sur une balise de titre, ou de paragraphe...

Par exemple, pour créer une ancre nommée monancre sur un titre, <h1 id="monancre">Ceci est mon titre</h1>

- préciser dans la balise <a> qu'on vise non plus une page en général, mais un endroit bien précis de cette page:

```
<!-- lien vers une ancre "monancre" sur la page en cours-->
<a href="#monancre">aller à mon ancre</a>

<!-- lien vers une ancre "monancre" sur une autre page-->
<a href="autrepage.html#monancre">aller à mon ancre sur une autre page</a>
```

1.2.4 balises sémantiques

Les balises div et span sont dites génériques. Elles peuvent être utilisées, mais les moteurs de recherche ne les interprètent pas.

HTML5 a donc introduit des balises sémantiques, à privilégier lorsque cela est possible.
Elles servent à délimiter des zones de la page:

- **<main>** partie principale de la page
- **<header>** Introduction (pour la page, une section, un article...
- **<footer>** pied de "page"
- **<nav>** barre de navigation (principale ou pour un article)
- **<section>** sections du document (chapitres...)

- `<article>` contenu indépendant, comme un article
- `<aside>` contenu additionnel
- `<details>` pour avoir une information que l'utilisateur peut afficher ou masquer. La structure est

```
<details>
<summary>Le sommaire, visible par défaut</summary>
<p> Le texte du détail</p>
<p> invisible par défaut</p>
</details>
```

- `<figure>` et `<figcaption>` pour définir une zone d'image et de légende:

```
<figure>
<img src="monimage.jpg" alt="mon image">
<figcaption>Fig1. - description de mon image</figcaption>
</figure>
```

Il est recommandé de donner un titre à chaque zone sémantique. Si vous ne souhaitez pas qu'il soit affiché, ne le rendez pas invisible (pour des raisons d'accessibilité), mais donner lui une taille de 1px par 1px;:

```
h1 {
   position: absolute;
   clip: rect(1px, 1px, 1px, 1px);
}
```

subsectionTexte

1.2.4.1 caractères spéciaux

Les balises utilisent les caractères < et >.
Si vous en avez besoin dans votre texte, il est donc impossible de les taper directement.
Pour obtenir:
< , on tape `<`
>, on tape `>`
& on tape `&`
Un espace insécable, on tape ` `
La symboles de copyright et trademark sont accessibles avec `&copy;`et `&reg;`.
Les symboles pour les euros `&euro;`, livres `&pound;`

Une liste plus complète est disponible à l'adresse: `https://dev.w3.org/html5/html-author/charref`.

1.2.4.2 Texte mis en valeur, en indice, exposant

Il existe différents niveaux de mise en valeur d'une partie de texte. Les plus utilisés sont:

- `<mark>` et `</mark>`pour une mise en évidence.
- `<em>` et `</em>`pour une emphase.

- `<strong>` et `</strong>` pour une forte emphase

Par défaut, la balise `<strong>` met le texte en gras et la balise `<em>` en italique.
Ne vous en servez pas pour cela.
Les moteurs de recherche se servent de ces balises pour déterminer ce qui est important sur la page.

Pour mettre en indice, on utilise la balise `<sub>` et `</sub>`, et pour mettre en exposant `<sup>` et `</sup>`.

1.2.4.3 Séparation sémantique

On utilise la balise `<hr>`, qui par défaut trace une ligne horizontale.

La modification du style de cet élément est possible en CSS.

Si une seule est présente, on peut modifier le style dans la balise: `<hr style="border-width:20px; margin-left:`

1.2.5 Listes

- liste non ordonnée:

  Une liste non ordonnée est délimitée par les balises `<ul>` et `</ul>` (unordered list)

  Complément: il est possible de changer le marqueur, à l'aide de `list-style-type` ou choisir une image comme marqueur.

  Dans le fichier CSS, on indique:

```
ul {
        list-style-image: url("monimage");
}
```

- liste ordonnée:

  Une liste ordonnée est délimitée par les balises `<ol>` et `</ol>` (ordered list).

  Par défaut, chaque élément sera précédé d'un nombre en chiffres arabes.

  On peut changer ce comportement à l'aide d'un attribut, c'est-à-dire d'un paramètre de la balise.

  On peut également changer le premier numéro, pour commencer la numérotation à une autre valeur, avec l'attribut `start`.

  L'attribut s'appelle ici `type`.

  Il prend pour valeurs:

  "1" pour le comportement par défaut.

  "I" ou "i" pour des nombres en chiffres romains, majuscules ou minuscules

  "A" ou "a" pour des lettres, en majuscules ou minuscules

- éléments d'une liste:

  Chaque élément d'une liste est délimité par les balises <li> et (list item)

- Exemples:

  Pour une liste non ordonnée:

```
<p> Ma liste</p>
<ul>
         <li>Mon premier élément</li>
        <li>Mon second élément</li>
</ul>
\end{html}

Pour une liste ordonnée, avec des chiffres romains minuscules:
\begin{minted}{html}
<p> Ma liste</p>
<ol type="I", start="3">
        <li>Mon premier élément</li>
        <li>Mon second élément</li>
</ol>
```

On peut imbriquer des listes.

N'oubliez pas les indentations pour ne pas vous perdre si vous décidez de mettre une liste dans une autre.

1.2.6 Liens

Parfois, on souhaitera qu'un lien s'ouvre dans un nouvel onglet. Pour cela, on utilise l'attribut `target`, avec la valeur `_blank`.

```
<!-- lien absolu vers la page d'accueil du site du lycée dans un nouvel onglet-->
<a href="http://www.lyc-jean-michel.ac-besancon.fr/" target="_blank">page
d'accueil du site du lycée</a>
```

Compléments

- mails: Si on souhaite créer un lien pour l'envoi de mails, on peut aussi utiliser une balise `<a>`.

  Il suffit de préciser dans l'attribut `href` une adresse mail commençant par `mailto:`

  Un clic sur le lien ouvrira le logiciel de messagerie de l'utilisateur.

  ```
  <!-- mail-->
  <a href="mailto:monnom@test.com" target="_blank">envoyer un mail</a>
  ```

- des fichiers à télécharger:

  Si l'adresse pointe vers un fichier qui ne peut s'afficher dans le navigateur, celui-ci sera téléchargé.

  ```
  <a href="http://www.monsite.com/fichier.pdf"> mon fichier</a>
  ```

  A noter, les navigateurs récents acceptent un attribut de la balise <a> qui permet de forcer le téléchargement au lieu de l'affichage:

  ```
  <!-- force le téléchargement d'un fichier html -->
  <a href="http://www.monsite.com/monfichier.html" download> mon fichier html</a>
  ```

```
<!-- force le téléchargement d'un fichier html en le renommant-->
<a href="http://www.monsite.com/monfichier.html" download="nouveaunom.html">
 mon fichier html</a>
```

- liens vers un endroit précis sur une page: Si votre page est suffisamment grande, vous ne souhaiterez probablement pas (et les utilisateurs encore moins) défiler jusqu'à l'endroit désiré.

  Pour créer des liens vers une endroit précis d'une page, il faudra:

  · créer une ancre, c'est-à-dire un point de repère sur votre page.
  Pour créer une ancre, on n'utilise pas une balise, mais un attribut, nommé id, sur une balise de titre, ou de paragraphe... Par exemple, pour créer une ancre nommée monancre sur un titre,

  ```
  <h1 id="monancre">Ceci est mon titre</h1>
  ```

  · préciser dans la balise <a> qu'on vise non plus une page en général, mais un endroit bien précis de cette page:

  ```
  <!-- lien vers une ancre "monancre" sur la page en cours-->
  <a href="#monancre">aller à mon ancre</a>

  <!-- lien vers une ancre "monancre" sur une autre page-->
  <a href="autrepage.html#monancre">aller à mon ancre sur une autre page</a>
  ```

1.3 Images

1.3.1 Miniatures

Si votre image est grosse, il peut être intéressant de n'afficher qu'une miniature, et de n'afficher la vraie image qu'en cas de clic:

Vous pouvez alors créer une miniature, à l'aide d'un logiciel d'édition d'image, puis

```
<a href="img/montagne.jpg"><img src="img/montagne_mini.jpg"
 alt="Photo de montagne" title="Cliquez pour agrandir"></a> </p>
```

1.3.2 Figures

Au lieu de placer une (ou plusieurs) image(s) dans un paragraphe, on peut la (ou les) placer dans une balise figure (qui indique à l'ordinateur de tenir compte de l'image)

```
<figure>
 <img src="images/monimage.jpg" alt="mon image">
 <figcaption>Ceci est mon image</figcaption>
</figure>
```

1.3.3 Images cliquables

Il est possible de définir des images cliquables en utilisant ismap ou usemap.

Ismap fait le travail côté serveur et usemap côté client. C'est une méthode plus simple et plus utilisée.

C'est donc celle que nous allons voir.

```
<img src="planets.gif" width="145" height="126" alt="Planets" usemap="#planetmap">
<map name="planetmap" id="mapmap">
  <area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun">
  <area shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury">
  <area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus">
  <area shape="default" href="page2.html" alt="le reste"/>
</map>
```

Pour un rectangle, x1,y1,x2,y2 bord gauche haut et bas droite Pour un cercle, x,y,r centre et rayon Pour un polygone: x1,y1,x2,y2,x3,y3 coordonnées des points.

En cas de chevauchement, les premières formes sont prioritaires.

1.3.4 Tableaux

Un tableau simple nécessite l'utilisation de 3 balises: la balise `<table>`, pour délimiter le tableau, la balise `<tr>` pour les lignes et `<td>` pour les colonnes.

```
<table>
 <!-- première ligne-->
 <tr>
  <td>L1C1 </td>
  <td>L1C2 </td>
 </tr>

 <!-- deuxième ligne -->
 <tr>
  <td>L2C1 </td>
  <td>L2C2 </td>
 </tr>
</table>
```

Pour un tableau un peu plus compliqué, on peut souhaiter avoir une légende, des en-têtes et pieds de tableau.

Pour la légende (ou le titre), on utilise la balise `<caption>`, juste après la balise `<table>` L'attribut caption-side, en CSS permet de changer la position de la légende, avec les valeurs: `top` ou `bottom`

Pour les en-têtes..., on sépare le tableau en 3:
L'en-tête, avec une balise `<thead>` Le corps avec une balise `<tbody>` Le pied, avec une balise `<tfoot>`

```
<table>
 <caption> Mon Titre</caption>
 <!-- en tête -->
 <thead>
  <tr>
   <td>Entête col 1</td>
   <td>En tête col2 </td>
  </tr>
 </thead>
```

```
 <!-- corps -->
 <tbody>
  <tr>
   <td>L1C1 </td>
   <td>L1C2 </td>
  </tr>
  <tr>
   <td>L2C1 </td>
   <td>L2C2 </td>
  </tr>
 </tbody>

 <!-- pied -->
 <tfoot>
  <tr>
   <td> Pied col1 </td>
   <td> Pied col2 </td>
  </tr>
 </tfoot>
</table>
```

Enfin, il est possible de combiner des cellules adjacentes. Si elles sont en ligne, on utilise l'attribut `colspan`avec pour valeur le nombres de colonnes occupées. Pour une cellule sur plusieurs lignes, on utilise `rowspan` avec comme valeur le nombre de lignes.

```
<table>
 <!-- première ligne-->
 <tr>
  <td>L1C1 </td>
  <td>L1C2 </td>
 </tr>

 <!-- deuxième ligne -->
 <tr>
  <td colspan="2">L2C1 </td>
 </tr>
</table>
```

On peut utiliser les CSS pour donner un style aux cellules, lignes, en-têtes... Pour des couleurs alternées, on peut, en CSS, utiliser:

```
tr:nth-child(even) { background-color: grey; }
tr:nth-child(odd) { background-color: lightgrey; }
```

1.3.5 Médias

1.3.5.1 Audio

Pour offrir la possibilité de lire un fichier audio, on utilise la balise `<audio>`.

Le texte écrit entre les balises d'ouverture et de fermeture sera affiché si le navigateur ne gère pas la balise audio.

Les attributs possibles sont:

- `src`, qui donne l'adresse du fichier source.

  La liste des formats disponibles dépend du navigateur, voir `https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats`

- `controls`, booléen qui permet l'affichage de boutons de contrôle
- `autoplay`, booléen qui lance la lecture dès que le fichier est prêt
- `loop`, booléen qui va lancer une répétition en boucle
- `muted`, booléen qui va couper le son
- `preload`, booléen qui va permettre d'automatiser le chargement du fichier. Les valeurs sont `auto` (charge audio et page en même temps), `metadata` (charge les données meta uniquement) et `none`.

Une présentation simple est alors: <audio src="sounds/son.mp3" controls autoplay> Votre navigateur n'est pas compatible. Veuillez le mettre à jour </audio>

Comme tous les formats ne sont pas gérés par les navigateurs, il est possible d'indiquer plusieurs fichiers sources, au cas où le premier ne serait pas lisible par l'utilisateur:

On obtient alors:

```
<audio controls preload="auto" autoplay>
 <source src="monfichier.mp3">
 <source src="monsecours.ogg">
</audio>
```

Dans un tel cas, si le navigateur ne peut pas lire le premier fichier, il lira le second. La balise source admet un attribut facultatif: le type, comme "audio/mpeg" ou "audio/ogg"...

1.3.5.2 Vidéos

Le fonctionnement de la balise `video` est similaire à celui de la balise `audio`, avec certaines caractéristiques de la balise `image`.

En effet, les attributs `src, controls, autoplay, loop, muted` et `preload` fonctionnent comme pour `audio`, de même que la possibilité de déclarer plusieurs fichiers sources (en général, en incluant un h264/mp4).

Il existe néanmoins quelques spécificités:

- Un attribut poster, qui définit l'adresse de l'image qui sera affichée avant lecture de la video (par défaut, c'est la première image de la vidéo).

  ```
  <video src="video.mp4" poster="monimage.png" controls>
  ```

- possibilité de redimensionner la video, en utilisant `width` et `height` sur le fichier CSS.

  Même si vous ne souhaitez pas redimensionner il est très intéressant de fixer les dimensions, pour que le navigateur attribue la bonne taille à la zone vidéo, pour éviter les déplacements ultérieurs du contenu.

- un attribut track, par exemple pour les sous-titres.

  Le standard est le format WebVTT.

  Il est possible d'en définir plusieurs:

```
<video src="mavideo.mp4" controls>
  <track src="sstitres/fr.vtt" label="Francais" default srclang="fr">
  <track src="sstitres/en.vtt" label="English" srclang="en">
</video>
```

1.3.5.3 Sites de partages

De nombreux sites de partages de vidéos (Youtube, Dailymotion...), de musique, mais aussi de cartographie, proposent d'intégrer leur contenu
Le code d'intégration à recopier dans la page est alors indiqué sur la page.

Chapter 2

CSS

L'acronyme CSS signifie Cascaded Style Sheets, soit feuilles de style en cascades.

L'objectif d'une feuille de style sera de modifier ou de préciser l'apparence souhaitée de portions de code html.

2.1 Choix des outils

2.1.1 Où placer les feuilles de style

Une feuille de style peut être placée à différents endroits. Même si pour des raisons de simplicité et de cohérence, nous n'en utiliserons qu'une, une rapide présentation s'impose:

- dans les balises du code de la page. `<p style="font-size: 14px">Mon texte</p>`

  Le fichier obtenu est en général plus difficile à lire.

  Il faut de plus coder chaque élément séparément.

  Nous n'utiliserons donc pas cette méthode.

  Malgré tout, en examinant des sites Web, vous trouverez ce type de codage. Cela ne signifie pas nécessairement que le créateur du site web a utilisé cette méthode, mais parfois qu'il y utilisé un langage de programmation appelé javascript pour injecter du code.

- dans le préambule du fichier, à l'aide d'une balise <style>.

  ```
  <head>
   <style>
    p {
        font-size:14px
      }
   </style>
  </head>
  ```

  Cette démarche est pratique, mais il est nécessaire de placer le code CSS dans chaque page. Pour un site comportant 100 pages, il faut donc 100 fois le même code.

  Comme pour la technique précédente, certaines méthodes automatiques injectent le code de cette manière.

- dans un fichier à part

  Si cette méthode peut sembler la plus lourde à gérer, c'est celle que nous utiliserons.

  En effet, un même fichier CSS pourra être utilisé pour déterminer le style de nombreuses pages.

  De plus, il sera facile de changer la présentation du site complet en modifiant un seul fichier.

  Il faudra simplement déclarer la feuille de style utilisée (ici monfichierstyle.css) dans l'entête de chaque page html, comme ceci (l'attribut type est maintenant facultatif en HTML5):

```
<!DOCTYPE html>
<html lang="fr">
 <head>
  <link rel="stylesheet" type="text/css" href="monfichierstyle.css">
```

  Remarque: il est tout à fait possible de mettre des informations de style à ces 3 endroits.

  C'est d'ailleurs une origine de l'utilisation de cascade. En effet, on peut définir des paramètres de style dans les 3 emplacements.
  S'ils sont compatibles, ils seront tous appliqués.
  Si ce n'est pas le cas, c'est celui défini en dernier qui le sera.

  Une autre raison est que les styles peuvent être transmis aux balises intérieures lorsqu'on les définit pour les balises extérieures (voir compléments CSS pour plus d'informations).

2.1.2 Production des fichiers

Comme pour les fichiers HTML, nous allons utiliser un éditeur de texte pour créer nos fichiers CSS.

Cette fois-ci, il faudra donner à votre fichier une extension .css

2.1.3 Vérification du code

Là encore, il sera indispensable de:

- valider votre code:

  Il existe pour cela des outils en ligne, comme `http://jigsaw.w3.org/css-validator/`

- tester, en pratique, sur différents navigateurs, y compris sur smartphone et tablette.

2.2 Premier fichier CSS

Rappelons le fichier HTML, auquel le lien vers le fichier CSS a été ajouté. La largeur de l'image, qui sera gérée dans le fichier CSS, a également été enlevée:

```
<!-- Mon premier fichier HTML -->
<!DOCTYPE html>
<html lang="fr">
 <head>
  <link rel="stylesheet" type="text/css" href="style1.css">
  <meta charset="utf-8" />
```

```
  <title>Une page un peu simplifiée</title>
 </head>

 <body>
  <!-- Contenu -->
   <h1>Mon titre Principal</h1>

   <p>Bienvenue sur ma page simplifiée.</p>
   <p> Si vous         cliquez sur les liens, vous n'irez nulle part.<br>
   Juste un petit passage à la ligne</p>

   <img src="http://www.lyc-jean-michel.ac-besancon.fr/sites/www.lyc-jean-michel/IMG/arton101.jpg
alt="logo du lycée">

  <!-- menu -->
   <ul class="navbar">
    <li><a href="#">Accueil</a></li>
    <li><a href="liens.html">Mes Liens</a></li>
   </ul>

 </body>
</html>
```

Comme pour la partie HTML, la première étape est d'ouvrir votre éditeur de texte préféré (ou celui qui est installé sur votre PC).
Créez un nouveau fichier et enregistrez le sous le nom style1.css
Ce fichier pourra être supprimé par la suite.

Dans un objectif de production, utilisez si possible des noms explicites, et évitez les fichiers nommés test.
Evitez aussi les caractères spéciaux, les espaces, les accents...

```
/* Fichier CSS:Un premier exemple
 * nom du fichier style1.css */

/* modification des paramètres de la police
 * des paragraphes */
p{
    font-size:12px;
    font-style:italic;
    font-family: "Times New Roman", Times, serif;
}

h1 {
    font-size:2em;
    font-style:italic;
    font-weight:bold;
}

img {
```

```
    width:40px;
    float:right;
}

a {
    color: #FF0000;
    background-color: blue;
}

ul {
    clear:both;
    border-width: 2px;
    border-color: green;
    border-style: solid;
}
```

Comme pour la partie HTML, n'hésitez pas à modifier des informations de ce fichiers, et regardez l'impact sur la page web associée.

2.3 Décryptage de l'exemple

Examinons l'allure générale du code: Vous pouvez remarquer que certaines informations ont été décalées vers la droite, ce qui n'a eu aucun effet sur la présentation de la page.

En effet, ce choix a été fait pour rendre le code plus lisible, mais le nombre d'espaces ou de retour à la ligne n'est pas important en HTML. Cette démarche est donc conseillée, mais pas obligatoire.

Vous observez que des blocs sont présents.

Chacun commence par une balise html, sans < et >.
Ensuite, on trouve une accolade ouvrante, puis une liste de mot-clé:valeur;.
Enfin, l'accolade est fermée.

Chaque élément HTML (p, body ...) sera appelé sélecteur.
Les mots-clés sont appelées propriétés.
Un couple mot-clé:valeur; est appelé une déclaration.

N'oubliez pas de terminer chaque déclaration par un point virgule.

Complément: Il est possible de modifier le style de plusieurs sélecteurs en même temps en les séparant par une virgule:

```
p, h1{
    font-style:italic;
}
```

Examinons maintenant ligne par ligne:

2.3.1 Commentaires

```
/* Fichier CSS:Un premier exemple
 * nom du fichier style1.css */
```

Comme pour le html, n'hésitez pas à commenter vos fichiers CSS, mais n'oubliez pas que vos commentaires sont visibles par tous les visiteurs du site.

En CSS, un commentaire débute par /* et se termine par */ Il peut être sur une ou plusieurs lignes.

2.3.2 Paramètres de police de caractères

```
p{
    font-size: 12px;
    font-style: italic;
    font-family: "Times New Roman", Times, serif;
}
```

Pour modifier la police de caractères utilisée (type, taille...)„ il existe une gamme de propriétés qui commencent toute par font-

- font-size

  Elle permet de modifier la taille de la police de caractère utilisée.

  La taille peut être précisée en taille absolue (12px pour 12 pixels de haut).

  C'est intéressant si vous savez sur quel écran la page va être vue, mais vous empêchez les redimensionnements par l'utilisateur.

  Elle peut aussi être exprimée en taille relative, c'est-à-dire par rapport à la taille par défaut spécifiée par l'utilisateur.

  On utilise pour cela l'unité em: 1em correspond la taille par défaut spécifiée par l'utilisateur. A noter, cette taille est relative à celle définie dans l'objet parent. Pour obtenir une taille relative à l'objet principal, on peut remplacer em par rem

  La recommandation du w3c est de rajouter, pour l'élément body, une taille fixée à 100 % , avant de modifier les éléments spécifiques, afin d'éviter les problèmes sur les anciens navigateurs.

  On obtient alors quelle chose qui ressemble à ceci:

  ```
  /* taille par défaut pour l'utilisateur: 16 px */
  body {
      font-size: 100%; /* pour les anciens navigateurs */
  }

  h1 {
      font-size: 2em; /* taille 16*2 = 32 px */
  }

  p {
      font-size: 0.75em; /* taille 16 * 0,75 = 12 px */
  }
  ```

  Enfin, une troisième possibilité est disponible, si on souhaite que la taille de la police varie en fonction de la largeur de l'écran (pour un site responsive): L'unité vw (viewport width). 1vw correspond à 1% de la largeur de l'écran.

Il existe de même vh (hauteur de l'écran).

Par exemple:

```
p {
    font-size: 3vw;
}
```

Un convertisseur est disponible à l'adresse : `http://pxtoem.com`

Complément: il est possible de faire des calculs, avec calc().
Par exemple, calc(100vh - 100px); La hauteur sera la hauteur totale de l'affichage - 100 pixels.

- font-style

  Elle permet de définir le style de la police: italique... Les valeurs possibles sont (en anglais): normal, italic, oblique, inherit.

  La valeur par défaut est normal.

  Inherit indique que la valeur sera reprise de l'élément parent.

- font-weight

  Elle permet de définir le poids de la police: gras...

  Les valeurs possibles sont (en anglais): normal, lighter, bold, bolder, inherit, ou 100, 200, 300, ..., 900.

  On peut donc choisir une valeur déterminée par un nom ou sur une échelle de 100 (la plus fine) à 900 (la plus épaisse), par pas de 100.

  La police la plus fine correspond à 100.

  La valeur par défaut est normal (soit 400).

  La police bold correspond à 700.

  Inherit indique que la valeur sera reprise de l'élément parent.

- font-family:

  Elle permet d'indiquer la police de caractère à utiliser si possible.

  En fonction des situations (différents systèmes d'exploitation, choix des utilisateurs...), les polices à disposition ne sont pas toujours les mêmes.

  La propriété font-family doit donc s'adapter à ce type de situations.

  Il sera donc possible de mettre plusieurs polices de caractères, séparées par des virgules.

  Si l'utilisation de la première est impossible, le navigateur essaiera la seconde, puis la troisième...

  Certaines polices sont considérées comme génériques, et à utiliser en fin de ligne, par sécurité: sans-serif (les plus utilisées, car les plus simples), serif (avec quelques décorations aux extrémités des caractères), monospace (tous les caractères ont même largeur. Elles ne sont pas les plus adaptées aux humains, et sont utilisées pour présenter des codes informatiques...), cursive (imitent l'écriture cursive humaine. A utiliser avec modération) et fantasy (polices décoratives).

  Une ligne classique pour cette description est donc

```
p {
    font-family: "Times New Roman", Times, serif;
}
```

N'oubliez pas les guillemets si le nom de votre police contient des espaces.

Une liste de lignes typiques est disponible à l'adresse family: `https://www.w3schools.com/cssref/css_websafe_fonts.asp`.

Une liste de police relativement sûres d'utilisation est disponible à l'adresse `https://www.cssfontstack.com/`.

Compléments:
Il existe également des polices disponibles sur le web, pour lesquelles il faut insérer une déclaration dans la page web.
Par exemple, des polices Google sont disponibles à l'adresse: `https://fonts.google.com/`.
Il faut choisir une ou plusieurs polices et insérer les lignes de code demandées avant de pouvoir utiliser les polices dans une page.

De la même manière, vous pouvez installer des polices sur votre site, disponibles par exemple à https://www.fontsquirrel.com

La même démarche permettrait de styliser les liens, les titres:

```
h1 {
    font-size:2em;
    font-style:italic;
    font-weight:bold;
}
```

2.3.3 Paramètres d'images

```
img {
    width:40px;
    float:right;
}
```

Il est possible de redimensionner, gérer le placement d'images.
Les propriétés communément utilisées sont:

- width et height

  Ils permettent de définir la largeur et la hauteur de l'image affichée.

- float

  Il indique que l'image va flotter. Les valeurs possibles sont left, right, none (par défaut) et inherit

- clear

  Il permet de préciser si des éléments peuvent flotter à gauche et à droite de l'élément considéré (on le met souvent à un paragraphe, pour éviter qu'une image ne flotte à ses côtés)

  Les valeurs prises sont none (par défaut, des éléments peuvent flotter à gauche et à droite), left (pas d'éléments flottants à gauche), right, both, inherit

2.3.4 Mettre en couleur

```
a {
    color: #FF0000;
    background-color: blue;
}
```

Il est possible de changer la couleur à l'aide de la propriété color, et la couleur du fond avec la propriété background-color

La valeur de cette propriété peut être:

- un nom (en anglais), parmi plusieurs possibilités, dont: black, blue, gray, green, navy, purple, red, white, yellow
- une valeur Rouge-vert-bleu, sous forme d'un triplet ou hexadécimale.

  Chaque couleur a alors une intensité entre 0 et 255 en décimal ou 0 et FF en hexadécimal.

  La notation est alors la suivante:

  ```
  /* pour du rouge */
  p {
      color: #FF0000; /* Rouge maximum, pas de vert et pas de bleu */
      /*color: RGB(255,0,0); */

      background-color: blue;
  }
  ```

  On peut trouver ces triplets à l'aide d'un logiciel de dessin. Un outil est même parfois disponible dans l'éditeur de texte utilisé (par exemple Geany).

  Compléments:
  La dernière notation permet également de gérer l'opacité du texte, en ajoutant un quatrième paramètre, entre 0 (invisible) et 1 (opaque):

  ```
  /* pour du rouge semi-transparent */
  p {
     color: RGBa(255,0,0,0.5);
  }
  ```

  Quelques points à noter:
  On ajoute a à la fin de RGB.
  Le séparateur décimal est le point et pas la virgule.

  Si on ne souhaite pas utiliser la notation RGB, mais qu'on désire de la transparence, on peut utiliser la propriété opacity.

2.3.5 bordures

Il existe des propriétés de gestion des bordures des éléments:

- border-width:

  Permet de définir la largeur de bordure.

La largeur peut être définie en px, em... ou en utilisant une valeur par défaut: thin, medium ou thick.

Comme pour border style, on précise soit une valeur, soit 4 valeurs distinctes, pour le haut, la droite, le bas et la gauche

```
/* pour des paragraphes encadrés en traits de largeur moyenne */
p {
   border-width: medium;
}

/* pour des paragraphes encadrés en traits de 2px sauf en bas: 5px */
p {
   border-width: 2px 2px 5px 2px;
}
```

- border-style:

  Permet de définir le style de bordure.

  Cette propriété prend les valeurs: dotted, dashed, solid, double, none, hidden. D'autres valeurs sont possibles, mais le résultat dépend de border-color: groove, ridge, inset, outset.

  Si une seule valeur est entrée, elle est appliquée aux 4 côtés.

  Il est également possible de préciser 4 valeurs distinctes, pour le haut, la droite, le bas et la gauche.

```
/* pour des paragraphes encadrés en traits pleins */
p {
   border-style: solid;
}

/* pour des paragraphes encadrés en traits pleins en haut, double en bas
et pointillés sur les côtés */
p {
   border-style: solid dotted double dotted;
}
```

- border-color:

  Le fonctionnement de cette propriété est identique à celui de la propriété color

- définition par côté

  Il est aussi possible de définir séparément les caractéristiques de chaque côté en remplaçant border par border-top, border-right, border-bottom ou border-left

- Raccourci

  Il est possible de compléter en une seule fois en utilisant la propriété border, et en complétant dans l'ordre largeur, style et couleur

```
/* pour des paragraphes encadrés en traits pleins,
   de largeur 2px et rouge */
p {
```

```
    border: 2px solid red;
}
```

On peut faire de même avec border-top, border-right...

Compléments: les tableaux.

Pour mettre en forme un tableau avec des cellules, il faut mettre un encadrement autour des cellules, et utiliser border-collapse sur table:

```
/* pour des cellules encadrées en
   traits pleins de largeur 2px et rouge */
table {
   border-collapse: collapse;
}

td{
   border: 2px solid red;
}
```

Nous sommes arrivés au bout de notre exemple. Il manque encore quelques paramètres fréquemment utilisés sur une page web

2.3.6 Autres paramètres importants

2.3.6.1 Les fonds d'écran

Il est possible d'insérer une image en fond d'écran pour un élément.

Si on souhaite appliquer ce fond d'écran à toute la page, il suffit de l'appliquer à la balise body

```
/* pour du rouge
body {
   background-image: url("monimage.jpg");
}
```

Il est possible de redimensionner cette image, de la placer à différents endroits ou de la répéter..

Il est également possible d'indiquer plusieurs images, qui seront affichées les unes à la suite des autres.

- redimensionner:

  On utilise la propriété background-size, avec deux valeurs: la largeur et la hauteur.

  Ces deux valeurs peuvent être absolues ou relatives.

  On peut également ne donner qu'une seule valeur. Dans ce cas, les proportions de l'image seront conservées.

- position sur la page.

  On utilise la propriété background-attachment pour déterminer si l'image va défiler en même temps que la page.

Les valeurs les plus utilisées pour ce paramètre sont scroll (par défaut, le fond va bouger avec le reste de la page), local (le fond va bouger avec le reste du contenu de son élément parent) et fixed (le fond d'écran va rester fixe)

- positionner.

  On utilise la propriété background-position, qui prend deux valeurs, qui correspondent à la position de l'image. On peut utiliser des valeurs absolues, relatives (en

- répéter.

  On utilise la propriété background-repeat, qui a pour valeurs possibles: repeat (par défaut), no-repeat, repeat-x (répéter horizontalement seulement), repeat-y (répéter verticalement seulement)

- plusieurs images.

  On peut également utiliser plusieurs images, séparées par des virgules:

```
/* pour du rouge
body {
    background-image: url("monfond1.jpg"), url("monfond2.jpg");
    background-repeat:repeat-x, no-repeat;
    background-size: 50%, 50%;
}
```

2.3.6.2 Mise en forme de texte: alignement, espacements, ombres...

Il est possible de gérer l'alignement...
On utilise pour cela les propriétés text-:

- text-decoration

  Permet de gérer le soulignement...

  Les valeurs prises par la propriété sont underline, overline, line-through, inherit, initial (valeur par défaut de la propriété), none

- text-align:

  Permet de gérer l'alignement.

  Les valeurs prises par la propriété sont left, right, center, justify, inherit

- text-indent:

  Permet de gérer les décalages vers la droite ou la gauche de la première ligne.

  Les valeurs prises par la propriété sont des nombres positifs ou négatifs, en valeurs absolues (px) ou relatives (em).

  Pour décaler l'ensemble du texte, on peut utiliser: padding-left.

- text-transform

  Permet de gérer le passage en minuscules, majuscules...

  Les valeurs prises par la propriété sont: lowercase, uppercase, capitalize (première lettre de chaque mot en majuscule), inherit, None (pour annuler un héritage par exemple).

- text-shadow:

  Permet de créer des ombres.

  La propriété prend 4 valeurs successives, séparées par des espaces:
  La projection horizontale, en px
  La projection verticale, en px
  Le rayon, en px
  La couleur (fonctionne comme color)

```
/* pour une ombre rouge en haut, à droite
p {
   text-shadow: 1px -1px 1px RGB(255,0,0);
}
```

Compléments:
Il est possible d'augmenter (ou de diminuer) les espacements entre les lettres et les espacement entre les mots, à l'aide des propriétés letter-spacing et word-spacing.

Les valeurs sont soit absolues, soit relatives, mais s'ajoutent à la valeur par défaut.

Il est également possible de changer les interlignes, à l'aide de line-height (qui prend des valeurs absolues ou relatives)

2.4 Projet 1: Etape 2 - différenciation des styles

On a pour l'instant donner un style identique à tous les paragraphes.

Il est possible de définir un style pour un seul paragraphe, ou pour un groupe précis.

2.4.1 Les sélecteurs class et id

La différence entre l'utilisation de ces deux sélecteurs est:

Si on souhaite donner un style à 1 seul élément, on utilise une id dans la fichier html.

Si on souhaite donner un style à plusieurs éléments, on utilise une class dans le fichier html

Par exemple:

```
<p id="parag1">Mon premier paragraphe</p>

<p class="paragsuivants">Mon second paragraphe</p>
<p class="paragsuivants">Mon troisième paragraphe</p>

<p>Mon quatrième paragraphe</p>
```

Dans ce fichier html, le premier paragraphe aura un style uniquement pour lui.
Nous avons donc utilisé l'attribut id.

Le second paragraphe et le troisième ont un style commun.
Nous avons donc utilisé un attribut class.

Enfin, le quatrième paragraphe utilisera le style habituel des paragraphes.

Les sélecteurs id et class étant associés à des noms choisis par l'utilisateur, la syntaxe dans le fichier CSS est différente.

```
/* Pour une id, on utilise # */
#parag1{
        font-size:13px;
}

/* Pour une class, on utilise un point. */
.paragsuivants{
               font-size:12px;
}
```

Comme vous pouvez le constater, un utilise une notation différente pour les class et id.
Il est donc possible, mais pas recommandé, de donner le même nom à une class et une id.

Compléments:
Avec cette notation, il est même possible de ne s'intéresser qu'à des éléments en fonction de conditions plus précises, ou moins précise:

- on souhaite gérer de manière séparée les éléments de type A inclus dans un élément B, par exemple les paragraphes inclus dans un div de classe "mondiv".

  ```
  .mondiv p{
      font-size:14px;
  }
  ```

- on peut également modifier uniquement les éléments HTML qui possèdent un attribut en particulier, voire même une valeur d'attribut.

  ```
  /* Change la taille des liens qui s'ouvriront dans un nouvel onglet*/
  a[target="_blank"]{
      font-size: 15px;
  }
  ```

  Si on ne précise pas de valeur d'attribut, les éléments modifiés seront tous ceux qui ont cet attribut, peu importe la valeur.

- Il est même possible de modifier tous les éléments en même temps, à l'aide du sélecteur *, ou simplement plusieurs sélecteurs en même temps:

  ```
  /* Donne la taille 14px aux paragraphes et aux titres h2*/
  p, h2{
      font-size:14px;
  }
  ```

2.4.2 Lettre mystère

On considère le code HTML suivant:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Projet 1</title>
    <meta charset="UTF-8" />
```

```
    <link rel="stylesheet" href="stylemystere.css">
  </head>

  <body>

    <p>
      <span>Ceci</span>
      <span>est</span>
      <span>un test</span>
    </p>

  </body>
</html>
```

Sauvegardez le sous le nom lettremystere.html Créer une feuille de style, avec le nom correct:

```
span
{
        display: inline-block;
        margin: 10px;
        box-shadow: 1px 2px 2px blue;
}
```

Modifier les 2 fichiers pour obtenir des mots stylisés différemment.
Correction possible:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Projet 1</title>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="stylemystere.css">
  </head>

  <body>

    <p>
      <span class="mot1 tournegauche">Ceci</span>
      <span class="mot2">est</span>
      <span class="">un test</span>
    </p>

  </body>
</html>
```

Sauvegardez le sous le nom lettremystere.html Créer une feuille de style, avec le nom correct:

```
span
{
        display: inline-block;
```

```
        margin: 10px;
        box-shadow: 1px 2px 2px blue;
}


.mot1 {
        background-color: yellow;
        font-family: "Times New Roman", Times, serif;
        font-weight: bold;
}

.mot2  {
        background-color: gray;
        color:white;
        text-transform: uppercase;
}

.tournegauche {
        transform: rotate(-10deg);
}
```

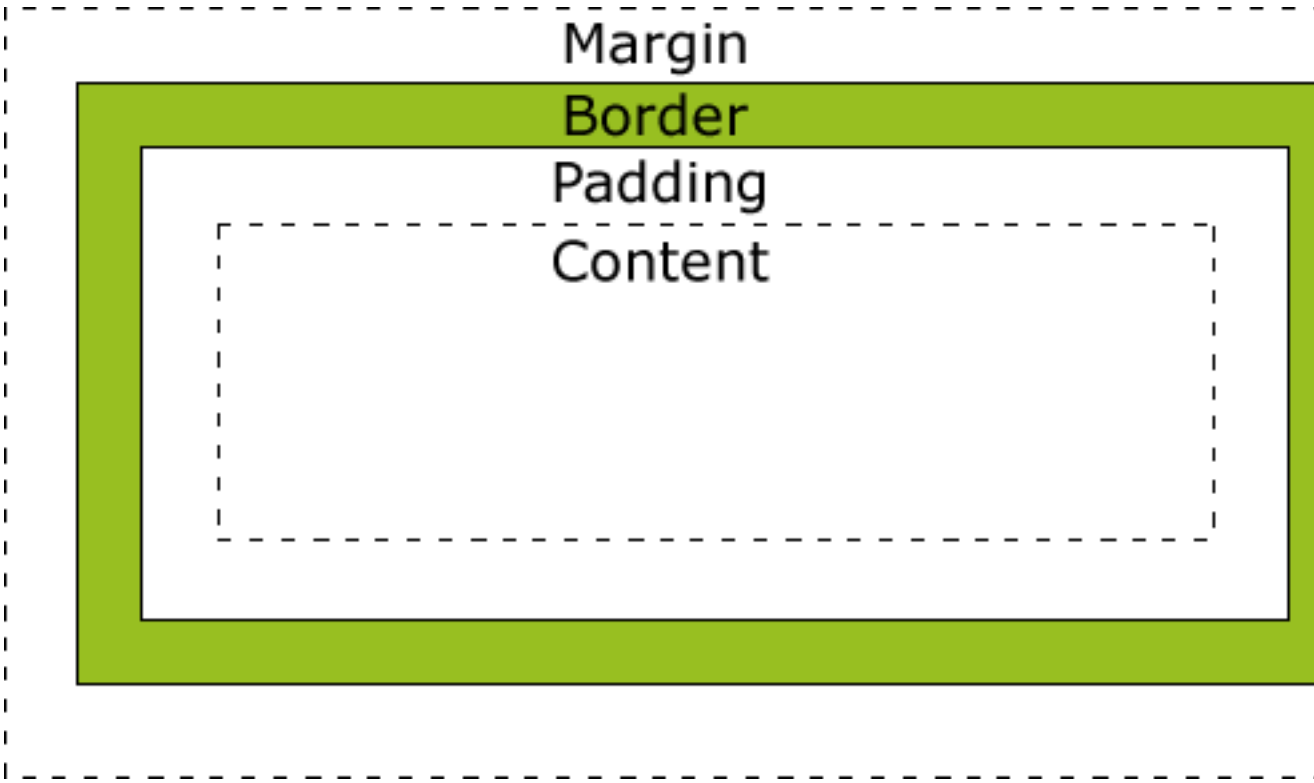
2.5 Projet 1: Etape 3 - placement du menu

En général, un menu est placé différemment sur une page web: en ligne tout en haut, ou sur la partie gauche, mais pas dans le flux.

2.5.1 Notion de boite

Les éléments HTML sont vus comme des boites, avec du contenu. La boite centrale représente le contenu de l'élément.

Autour, on trouve la marge intérieure (padding), puis la bordure (border), puis la marge extérieure (margin).



On peut régler indépendamment les dimensions du padding, border et margin. La démarche a déjà été vue dans les bordures.

Pour la couleur: le padding est à l'intérieur. Sa couleur est donc la couleur de fond de l'élément. La couleur de la bordure peut être réglée (déjà vu). La marge extérieure est à l'extérieur. Sa couleur est donc déterminée par la couleur des éléments extérieurs.

Pour les ombres: on peut créer une ombre autour d'une boite comme on a créé une ombre autour d'un texte.
La commande est box-shadow.

On pourrait donc placer le menu dans une boite, et le contenu de la page dans une autre.

Un solution déjà connue est de mettre le menu dans une div: et le contenu dans une autre.

C'est une solution que vous trouverez sur de nombreux sites, et qui fonctionne très bien.

Il existe néanmoins des boites déjà prévues en HTML5:

- <header>: pour l'en-tête
- <nav> pour la barre de navigation
- <section> pour les sections
- <article>, pour chaque article
- <aside> zone secondaire
- <footer> pied de page

Elles ne sont pas obligatoires, mais sont utilisées par certains moteurs de recherche et rendent plus faciles la compréhension de la structure de la page.

Que vous choisissiez les balises intégrées à HTML5 ou des div, l'avantage est que les boites peuvent être déplacées dans le code CSS:

2.5.2 position

Il est possible de gérer le positionnement d'une boite à l'aide de la propriété position, et des propriétés top, left, right, bottom. Les valeurs possibles sont static (ne peut pas être affecté par top...), relative (repositionné), fixed (on choisit la position), absolute (par rapport à un parent qui a été positionné à l'aide de relative, fixed ou absolute ou hml par défaut).

```
p {
   position: relative;
   left: 30px; /* déplacé de 30 px vers la droite */
}
```

Ou

```
p {
   position: fixed;
   top: 10px;
   left: 30px; /* à 10px du haut et 30 de la gauche */
}
```

Remarque: En déplaçant ainsi des objets, il est possible que certains se chevauchent.

Pour choisir quel élément va apparaitre au-dessus, on utilise la propriété z-index.

Celle-ci prend pour valeurs des nombres entiers. Plus le nombre est grand, plus l'élément est à l'avant. Compléments:

2.5.2.1 position relative

La boite va être déplacée par rapport à sa position initialement prévue: position: relative; top: 50px; left: 100px; /* déplace de 100 px vers la droite et 50 vers le bas.

La boite peut venir au-dessus ou en-dessous d'une autre boite, mais sa place initiale est conservée (et donc potentiellement vide)

2.5.2.2 position absolue

La boite va être déplacée par rapport à son objet parent: position: absolute; top: 50px; left: 100px; /* place à 100 px vers la droite et 50 vers le bas du coin haut gauche de l'objet parent.

La boite peut venir au-dessus ou en-dessous d'une autre boite, mais sa place initiale est perdue, et occupée par le reste du document, comme si la boite n'existait pas

2.5.2.3 position fixed

La boite ne bougera pas, même en cas de défilement dans la page. position: fixed; top: 50px; left: 100px;

La boite peut venir au-dessus ou en-dessous d'une autre boite, mais sa place initiale est conservée (et donc potentiellement vide).

Attention, les déplacements en pourcentages ne sont pas interprétés de la même manière selon le positionnement.

En relative, un déplacement de 50% correspond à 50% de la boite, alors qu'en absolute, c'est 50% de l'objet parent.

En cas d'informations incompatibles, c'est la dernière qui est appliquée.

2.5.3 Notre exemple

En utilisant les balises nav et section, notre fichier html devient

```
<!DOCTYPE html>
<html lang="fr">
 <head>
  <meta charset="utf-8" />
  <link rel="stylesheet" type="text/css" href="style1.css">
  <title>Une page un peu simplifiée</title>
 </head>

 <body>
  <!-- Contenu -->
  <section>
   <h1>Mon titre Principal</h1>

   <p>Bienvenue sur ma page simplifiée.</p>
   <p> Si vous          cliquez sur les liens, vous n'irez nulle part.<br>
   Juste un petit passage à la ligne</p>

   <img src="http://www.lyc-jean-michel.ac-besancon.fr/sites/www.lyc-jean-michel/IMG/arton101.jpg
   alt="logo du lycée" width="40">
  </section>

  <!-- menu -->
  <nav>
   <ul class="navbar">
    <li><a href="#">Accueil</a></li>
    <li><a href="liens.html">Mes Liens</a></li>
   </ul>
  </nav>
 </body>
</html>
```

On peut alors placer le menu en ajoutant les sélecteurs nav et section:

```
section {
    position:absolute;
    left:15%
}

nav {
```

```
    width:15%;
    position:absolute;
    left:0%
}
```

On pourrait aussi utiliser float et des marges

2.5.4 Menu horizontal

Les éléments de liste sont du type block.

Il existe une propriété CSS qui permet de changer cet état de fait: la propriété display.

```
/* pour des paragraphes en ligne
p {
   display: inline;
}
```

Cette propriété prend les valeurs: inline, block, mais aussi inline-block et none (aucun affichage)

Pour notre menu, on va utiliser inline-block: les éléments inline-block sont positionnés les uns à côté des autres (comme inline), mais peuvent être redimensionnés (comme des blocks).

```
nav {
    display: inline-block;
    vertical-align: top;
    width: 15%;
}

section {
    display: inline-block;
}
```

On peut utiliser la même technique pour créer un menu horizontal. Inline-block ne s'appliquera plus à section et nav, mais à li

2.5.5 Menu horizontal

```
ul {
    list-style: none; /* pas de puce */
}

li
{
width: 150px; /* taille des éléments de la liste. */
display: inline-block;
}
```

Améliorations: On pourrait

- modifier la mise en forme du texte, le soulignement...

- mettre les liens sur toute la largeur de l'élément li, des marges...

```
li a {
        display: block;
        width: 100%;
        height: 100%;
}
```

- Changer la couleur du fond d'un lien en cas de survol. Il est possible de définir des styles différents pour les liens en fonction de leur état:

```
a:link /* non visité */
a:visited /* visité */
a:active /*actif */
a:hover /* survol */
/* hover et active peuvent être utilisés sur des éléments autres que des liens

p:hover
```

Exemple: modifier un style au passage de la souris:

```
/* Style par défaut */
p
{
 transition-property: color, height;
 transition-duration: 2s;
 transition-delay: 1s; /* débute après 1s */

 height: 100px;
 color: black;
 text-align: center;
}

/* Style au survol de la souris. */
p:hover
{
 height: 150px;
 color: blue;
}
```

Le fichier CSS complet pourrait alors être:

```
p{
    font-size:12px;
    font-style:italic;
    font-family: "Times New Roman", Times, serif;
}

h1 {
    font-size:2em;
```

```
    font-style:italic;
    font-weight:bold;
}

img {
    width:40px;
    float:right;
}

a {
    color: #FF0000;
    background-color: blue;
}

ul
{
list-style: none;
}

li
{
width: 150px; /* On fixe les dimensions des éléments de notre liste. */

display: inline-block;
text-align: center;
margin: 15px; /* éloigne un peu les éléments les uns des autres */
border: 1px solid yellow;
border-radius: 10px;
Overflow:hidden; /* n'affiche pas ce qui dépasse */
}


li a
{
/* notre lien occupera tout son élément li. */
display: block;
width: 100%;
height: 100%;
text-decoration: none;
}

a:hover
{
background-color:gray;
}
```

2.6 Animation

Notre objectif est d'obtenir une animation.

On considère pour cela le fichier HTML suivant:

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title>Animation</title>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div id="ciel">
        <img id="soleil" src="soleil.jpg" alt="soleil" />
    </div>

    <div id="mer">
    </div>
  </body>
</html>
```

2.6.1 ciel et mer

On souhaite colorer le ciel en bleu clair, la mer en bleu, que le ciel et la mer occupent chacun la moitié de la page.

On obtient alors le CSS suivant:

```
body {
  margin: 0;
}

html {
  overflow: hidden;
}

#ciel {
  position: absolute;
  width: 100%;
  height: 50%;
  background: lightblue;
}

#mer {
  position: absolute;
  width: 100%;
  height: 50%;
  bottom: 0;
```

```
  background: blue;
}
```

2.6.2 Placer le soleil

Il faut ajouter dans le fichier CSS:

```
#soleil {
  position: absolute;
  top:0;
  left: 50%;
  height: 100px;
}
```

2.6.3 Animer le soleil

Cette partie est nouvelle:

```
@keyframes mouvsoleil {
        0% {top: 90%;}
        100% {top: 0;}
}
```

Ce code indique au soleil où se positionner au début (0%) et à la fin (100%)de l'animation. Puisque le soleil est à l'intérieur de la div ciel, les positions top et left que vous donnez sont comprises dans le ciel, avec top : 100% étant le bas du ciel, et non le bas de la page Web.

Pour l'instant, l'animation est définie, a un nom, mais n'est pas appliquée.

Pour cela, dans # soleil, ajouter: animation: mouvsoleil 10s;

Maintenant, modifier cette animation pour que le soleil se lève, arrive au zenith, puis se couche:

```
@keyframes mouvsoleil {
0% {top:90%; left:0;}
33% {top:0; left:40%; } 66% {top:0; left:40%; } 100% {top:90%; left:80%; }
}
```

L'animation est pour l'instant jouée une fois. Pour qu'elle se répète, il faut modifier l'appel à l'animation dans # soleil, en ajoutant infinite. On obtient: animation: sunrise 10s infinite;

2.6.4 Animer le ciel

En l'état, que le soleil soit couché ou au zenith, le ciel a la même couleur.

Créez une nouvelle animation pour que le ciel change de couleur. @keyframes sky 0

2.7 Hiérarchie

Les éléments HTML sont liées par une hiérarchie: un élément directement dans un autre est un enfant. Celui directement à l'extérieur est un parent.

L'analogie se poursuit de même pour les ancêtres, descendants, fratries...

Par défaut, une propriété d'avant-plan est héréditaire et une propriété d'arrière-plan ne l'est pas. Si une propriété héréditaire est définie pour un parent, elle s'applique à tous ses enfants, sauf si cette propriété a été modifiée pour un enfant.

```
/* séparés par des espaces */
bal1 bal2 /* pour les bal2 descendants de bal1 (peu importe le nombre de niveaux les séparant) */

/* séparés par > */
Bal1 > bal2 /* est pour les bal2 enfants de bal1 (descendants directs) */



/* Pour les fratries: */
/*séparés par + */
h1 + p /* pour les paragraphes immédiatement après un titre h1 */

/* séparés par ~ */
h1 ~ p  /* pour les paragraphes après un titre h1, même si ce n'est pas immédiatement après,
               mais dans une même balise parent */
```

2.8 Les dégradés de couleurs

Il existe deux types de dégradés:

- les dégradés linéaires

  ```
  #body {
    /* dégradé de gauche à droite, du rouge au bleu */
    background: linear-gradient(to right, red , blue);
  }
  ```

  On précise dans la fonction linear-gradient la direction (qui peut aussi être en angle, en degrés), puis une liste de couleurs.

  Exemple plus complet:

  ```
  #body {
    /* dégradé avec un angle de 45°, passant du rouge à l'orange, puis au jaune, puis au bleu*
    background: linear-gradient(45deg, red , orange, yellow,blue);
  }
  ```

  Il est possible de préciser, sur une échelle de 0% (début) à 100% (fin), à quel endroit on doit trouver une couleur pure.

  ```
  #body {
    /* dégradé avec un angle de 45°, passant du rouge à l'orange, puis au jaune, puis au bleu*
    background: linear-gradient(red 5%, orange 20%, yellow 40%,blue 80%);
  }
  ```

- les dégradés "circulaires"

Il est possible de créer des dégradés elliptiques ou circulaires. La fonction à utiliser est radial-gradient.

Par défaut, le dégradé est elliptique. Pou obtenir un dégradé circulaire, on indique un premier paramètre circle.

```
#body {
  /* dégradé circulaire, passant du rouge à l'orange, puis au jaune, puis au bleu*/
  background: radial-gradient(circle, red , orange, yellow,blue);
}
```

2.9 Alignement

2.9.1 Centrer

Les méthodes vont varier selon le type d'objet à centrer:

- dans un élément block:

  Pour centrer le contenu d'un élément block, on utilise margin:auto; Ou margin-left:auto; margin-right:auto; En générale, on définit en même temps la largeur du bloc.

- dans un élément hors du flux:

  Méthode1:

```
<div id="wrapper">
 <ul id="menu">
  <li><a href="#">Premier lien</a></li>
  <li><a href="#">Deuxième</a></li>
  <li><a href="#">Troisième</a></li>
 </ul>
</div>

#wrapper {
  /* masque les debordements et englobe le menu */
  overflow: hidden; }

#menu, #menu li {
/* on supprime les eventuels styles par défaut */
margin: 0;
padding: 0;
list-style: none;

/* float permet à l element de prendre la largeur de son contenu */
position: relative; }

#menu {
/* on decale le menu vers la droite de la moitie de la largeur disponible */
left: 50%; }
```

```
#menu li {
/* on décale chaque item vers la gauche de la moitié de la largeur du menu */
right: 50%; }
```

Méthode 2: préciser la largeur, le décaler de 50 % vers la droite, puis lui imposer une marge négative de la moitié de sa largeur

```
#element {
Width:400px;
Margin-left:-200px;
Position:absolute;
Left:50%
}
```

2.9.2 Alignement vertical

```
Pour les objets inline-block, on utilise vertical-align
vertical-align:baseline; /* bases de d'élément et du parent alignées */
vertical-align:top; /* aligné en haut */
vertical-align:middle; /* centre verticalement */
vertical-align:bottom; /* aligné en bas */
vertical-align: (valeur en px ou \%); /* pourcentage par rapport à la base du parent */
```

2.10 Overflow

Cette propriété contrôle ce qui va se passer en cas de dépassement de taille du contenu.

Les valeurs possibles sont visible (le contenu déborde), hidden (contenu supplémentaire invisible), scroll (barres de défilement) et auto (le navigateur choisit).

2.11 Menu déroulant

code html

```
<nav>
        <section>
                <h3>Bloc1</h3>
                <ul>
                        <li><a href="#">el1</a></li>
                        <li><a href="#">el2</a></li>
                        <li><a href="#">el3</a></li>
                </ul>
        </section>
        <section>
                <h3>Bloc2</h3>
                <ul>
                        <li><a href="#">el1</a></li>
                        <li><a href="#">el2</a></li>
                        <li><a href="#">el3</a></li>
```

```
                        <li><a href="#">el4</a></li>
                </ul>
        </section>
</nav>
```

Code CSS

```
nav {
        text-align: center;
}

nav section {
        display: inline-block;
        vertical-align: top;
}

nav ul {
        display: none;
        position: absolute;
        padding: 0;
        margin: 0;
        list-style: none;
}

nav section:hover ul {
        display: block;
}

nav h3, nav a {
        display: block;
        margin: 0;
        padding: .5em 1.5em;
        font-size: inherit;
        text-decoration: none;
        box-sizing: border-box;
        width: 10em;
}

nav section:hover h3 {
        color: white;
        background-color: steelblue;
}
```