INTRODUCTION A HTML5 ET CSS3

I) LES BASES DE HTML5 :

Le HTML5 et le CSS3 sont les dernières évolutions

1) COMMENT FONCTIONNENT LES SITES WEB ?

Pour consulter un site web vous lancez un navigateur : Firefox, Internet-explorer, Chrome…
Le HTML (inventé en 1991) et le CSS sont des langages informatiques qui permettent de créer des sites web.
Quand vous consultez un site web, le navigateur interprète le code HTML/CSS de la page et l'affiche.

Code source de la page web

Interprétation par la navigateur

Résultat à l'écran

2) ROLES DE HTML ET CSS :

- HTML (HyperText Markup Language) : il a fait son apparition dès 1991 lors du lancement du Web. Son rôle est de gérer et organiser le contenu. C'est donc en HTML que vous écrirez ce qui doit être affiché sur la page : du texte, des liens, des images…

- CSS (Cascading Style Sheets, aussi appelées Feuilles de style) : le rôle du CSS est de gérer l'apparence de la page web (agencement, positionnement, décoration, couleurs, taille du texte…). Ce langage est venu compléter le HTML en 1996.

Comme on le voit Le CSS permet, d'arranger le contenu et de définir la présentation : couleurs, image de fond, marges, taille du texte…

3) CREATION D'UN SITE WEB :

Pour créer un site web, il existe des logiciels puissants, cependant un gros défaut est la qualité souvent assez mauvaise du code HTML et CSS qui est automatiquement généré par ces outils.
Pour créer un site web « *propre* » , un simple éditeur de texte suffit : par exemple notepad++ http://notepad-plus-plus.org/fr/
Il a l'avantage de colorer le code pour s'y retrouver plus facilement.

4) LES NAVIGATEURS :

Le navigateur est le programme qui nous permet de voir les sites web. Cependant le rendu visuel d'un site peut ne pas être le même suivant celui que vous utilisez. Il faut donc vérifier que le site s'affiche correctement sur plusieurs navigateurs. On peut par exemple tester la compatibilité avec les versions de IE, avec IETester
http://www.mydebugbar.com/wiki/IETester/HomePage
De plus maintenant, il faut vérifier l'affichage sur les mobiles, tablettes….
On peut tester un site sur différents navigateurs:
https://www.browserling.com/
https://www.browserling.com/

II) PREMIERE PAGE EN HTML :

Pour créer une page web, il ne suffit pas de taper simplement du texte et de l'enregistrer en .html. En plus de ce texte, il faut aussi écrire ce qu'on appelle des **balises (**aller à la ligne, afficher une image, lien, …)
Les balises sont invisibles à l'écran, elles permettent à l'ordinateur de comprendre ce qu'il doit afficher.
Les balises se repèrent facilement. Elles sont entourées de « chevrons », c'est-à-dire des symboles < et >, comme ceci : <balise>.
Il y a les balises en paires et les balises orphelines.
Exemples :

- ⇨ Balises en paires (ouvrantes et fermantes) :
  <titre>ceci est un titre**</titre>**
- ⇨ Balise orpheline :
  <img />

Les balises peuvent également contenir des attributs : Ils viennent les compléter pour donner des informations supplémentaires.
<balise attribut= "valeur">expressions**</balise>**

1) SRUCTURE DE BASE D'UNE PAGE HTML :

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>Titre</title>

   </head>

   <body>

   </body>
</html>
```

On décale les balises pour que cela soit plus clair

Attention à ne pas entremêler les balises. Ex : <html><body></html></body>

<!DOCTYPE html>: elle qui indique qu'il s'agit bien d'une page web HTML5 (beaucoup plus simple qu'avant)
<html> : englobe tout le contenu de la page
<head> : balise d'en-tète (informations générales sur la page => ne s'affiche pas à l'écran)
<title> : titre de la page (affiché par exemple au niveau des moteurs de recherche)
<meta charset="utf-8" />: Cette balise indique l'encodage utilisé dans votre fichier (affiche sans problème toutes les langues)
Attention à bien enregistrer le fichier en utf-8
<body> : partie principale de la page (s'affiche à l'écran)

On peut également insérer des commentaires qui servent de mémos : **<!--** Ceci est un commentaire **-->**

Tout le monde peut voir le code HTML de votre page une fois celle-ci mise en ligne sur le Web.

- ⇨ Ne mettez donc pas d'informations sensibles (mots de passe …)

2) ORGANISER SON TEXTE :

Il est nécessaire de structurer sa page convenablement pour la rendre plus attractive. On rappelle que le contenu s'écrit entre les balise <body> et </body>.
Lorsque l'on écrit du texte, on l'organise en titres éventuellement sous-titres , paragraphes et listes.

- ⇨ Balises titres du plus important au moins important :
  <h1>... </h1>; <h2>... </h2>; <h3>... </h3>; <h4>... </h4>; <h5>... </h5>; <h6>... </h6>
- ⇨ Balises de paragraphes :
  ** …. **
- ⇨ Aller à la ligne :
  **
**
- ⇨ Mettre en valeur du texte :
  …
  …
  <mark>…</mark>

⇨ Listes ordonnée ou non :
**
…
** délimite toute une liste non-ordonnée ;
**
…
** délimite toute une liste ordonnée ;
**
…
** délimite un élément de la liste (une puce).

Exemple :
```
<ol>
   <li>Je me lève.</li>
   <li>Je mange et je bois.</li>
   <li>Je retourne me coucher.</li>
</ol>
```

On pourra bien sur changer la mise en forme se fera grâce au langage CSS

Il existe également des balises universelles (elles n'ont pas de significations particulières, on ne sait pas ce que l'on a, elles sont utilisées de manière combinée avec du CSS):

⇨ **… ** : c'est une balise de **type inline**, c'est-à-dire une balise que l'on place au sein d'un paragraphe de texte, pour sélectionner certains mots uniquement. Les balises et sont de la même famille..
Les élément de type inline **prennent uniquement la largeur qui leur est nécessaire** (c'est-à-dire la largeur de leur contenu).

⇨ ** … ** : c'est une balise de **type block**, qui entoure un bloc de texte. Les balises <p>, <h1>, etc. sont de la même famille. Ces balises ont quelque chose en commun : elles créent un nouveau « bloc » dans la page et provoquent donc obligatoirement un retour à la ligne. est une balise fréquemment utilisée dans la construction d'un design. Elles possèdent **une largeur**, une marge **intérieur** et **extérieur** ..
Pour les éléments de type block , on pourra modifier la largeur et pleins d'autres propriétés, ….
Par défaut, **elles prennent toute la largeur disponible dans leur élément parent.**

---

3) CREER DES LIENS :

⇨ **titre du lien :** créer un lien absolu vers un autre site
⇨ **titre du lien :** créer un lien relatif vers une page du site
⇨ **titre du lien :** créer un lien relatif vers une page du site
⇨ **titre du lien :** créer un lien relatif vers une page du site se trouvant dans le dossier parent. (../ on remonte l'arborescence)
⇨ Lien vers une ancre :
On utilise l'ancre lorsque la page est très longue
- **<h2 id="mon_ancre">Titre</h2>** : création de l'ancre
- **Aller vers l'ancre** : lien vers l'ancre
- **texte** : aller vers une ancre dans une autre page

⇨ Quelques attributs (infos complémentaires) de la balise <a> :
- **titre du lien :** affichage d'une infobulle
- **titre du lien :** ouverture du lien dans une nouvelle page ou onglet selon le navigateur.

⇨ **Envoyez un e-mail !** : pour envoyer un e-mail
⇨ **Télécharger le fichier** : télécharger un fichier.

---

4) LES IMAGES :

Il existe plusieurs formats : (attention au poids et dimensions => temps de chargement)
Si vous avez :
⇨ Une photo : utilisez un JPEG.
⇨ N'importe quel graphique avec peu de couleurs (moins de 256) : utilisez un PNG 8 bits ou éventuellement un GIF.
⇨ N'importe quel graphique avec beaucoup de couleurs : utilisez un PNG 24 bits.
⇨ Une image animée : utilisez un GIF animé.

Evitez de mettre des accents et caractères spéciaux dans le nom des fichiers.

⇨ ** :** insertion d'une image dans une page

5) <u>LES BALISES STRUCTURANTES DE HTML5 :</u>

Tout document HTML5 dispose de cloison de contenu que sont article, aside, nav et section. Ces zones de contenu peuvent chacune contenir une balise header et footer (ne cloisonnant pas elles-mêmes le contenu) et de multiple éléments de titrage allant de h1 à h6.

Avant on avait des balises de type bloc ** ** , mais elles n'avaient aucune signification au niveau de la sémantique

Donc on a introduit en HTML5 d'autres balises de type bloc qui apportent une précision au niveau de la sémantique (dédiées à la structuration du site)

Balise **<header>** : spécifie l'entête du site web

Balise **<aside>** : donne des infos supplémentaires

Balise **<nav>** : menu

Balise **<hgroup>** : regroupe un ensemble de titres (<h1> , <h2>, …)

Balise **<section>** : Section générique regroupant un même sujet, une même fonctionnalité

Balise **<article>** : doit contenir un article : une news, un message (et un seul)

Balise **<footer>** : pied de page

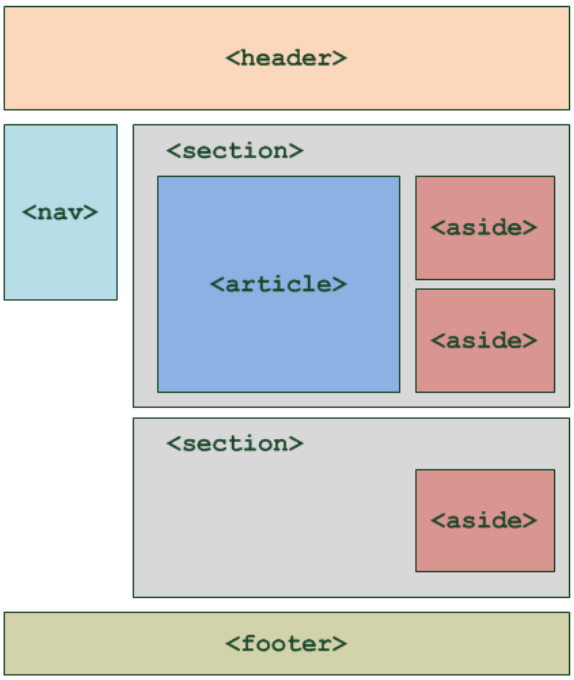
En général, une page web est constituée d'un en-tête (tout en haut), de menus de navigation (en haut ou sur les côtés), de différentes sections au centre… et d'un pied de page (tout en bas). On pourrait très bien utiliser des balises génériques

- **<header>** : l'en-tête

```
<header>
   <!-- Placez ici le contenu de l'en-tête de votre page -->
</header>
```

- **<footer>** : le pied de page
- …

Exemple de structure schématique d'une page qui n'est pas unique



Les éléments **<figure>** et **<figcaption>** fonctionnent de concert pour associer une légende à une illustration ou un autre élément média (images, des schémas, des vidéos, des tableaux ou encore des blocs de code) (figure instaure des marges par défaut à éventuellement corriger)

```
<figure>
  <img src="image.jpg" alt="" />
  <figcaption>Légende associée</figcaption>
</figure>
```

Les nouvelles balises ne sont reconnues par Internet Explorer que depuis sa version 9 (IE9).

Il existe un script qui permet de faire en sorte que (<header>, <footer>, <section>…) s'affichent correctement sur les anciennes versions d'Internet Explorer (IE6, IE7, IE8).

Il faut rajouter les lignes suivantes dans le code :

```
<!--[if lt IE 9]>
   <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

6) LES TABLEAUX :

a) Tableaux basiques :

- ⇨ **<table> </table>** : Permet d'indiquer le début et la fin d'un tableau.
- ⇨ **<tr> </tr>** : indique le début et la fin d'une ligne du tableau ;
- ⇨ **<td> </td>** : indique le début et la fin du contenu d'une cellule.
- ⇨ **<th> </th>** : indique les cellules d'entête.
- ⇨ **<caption> </caption>** : titre du tableau

```
<style>
table
{
border: 1px solid black;
border-collapse: collapse;
/* pas d'espacement entre les bordures des cellules ( separate ou collapse)
*/

}

td, th
{
border: 1px solid black;
}
</style>

<table>
  <caption>Etudiant BTS SN</caption>
  <tr>
    <th>Nom</th>
    <th>Âge</th>
    <th>Adresse</th>
  </tr>
  <tr>
    <td>Nom 1</td>
    <td>Age 1</td>
    <td>Adresse 1</td>
  </tr>
  <tr>
    <td>Nom 2</td>
    <td>Age 2</td>
    <td>Adresse 2</td>
  </tr>
</table>
```

b) Tableaux structurés :

Pour les gros tableaux, on peut éventuellement utiliser les balises :

- ⇨ **<thead></thead>** : l'en-tête (en haut)
- ⇨ **<tbody></tbody>** : le corps (au centre)
- ⇨ **<tfoot></tfoot>:** le pied du tableau (en bas) , ici on recopie en général l'entête

```
<table>
  <caption>Etudiant BTS SN</caption>
<thead> <!-- En-tête du tableau -->
  <tr>
    <th>Nom</th>
    <th>Âge</th>
    <th>Adresse</th>
  </tr>
</thead>

<tfoot> <!-- Pied de tableau -->
  <tr>
```

```
    <th>Nom</th>
    <th>Âge</th>
    <th>Adresse</th>
  </tr>
</tfoot>

<tbody> <!-- Corps du tableau -->
  <tr>
    <td>Nom 1</td>
    <td>Age 1</td>
    <td>Adresse 1</td>
  </tr>
  <tr>
    <td>Nom 2</td>
    <td>Age 2</td>
    <td>Adresse 2</td>
  </tr>
</tbody>
</table>
```

c) <u>Fusions de cellules :</u>

On doit rajouter un attribut à la balise <td>

- ⇨ **colspan** : fusion de colonnes : La fusion s'effectue horizontalement.
- ⇨ **rowspan** : La fusion de lignes : La fusion s'effectuera verticalement.

```
<style>
table
{
border: 1px solid black;
border-collapse: collapse;
}

td, th
{
border: 1px solid black;
}
</style>

<table>
  <caption>Etudiant BTS SN</caption>
  <tr>
    <th>Nom</th>
    <th>Age</th>
    <th>Adresse</th>
  </tr>
  <tr>
    <td>Nom 1</td>
    <td rowspan="2">Age 1</td>
        <td>Adresse 1</td>
  </tr>
  <tr>
    <td>Nom 2</td>

    <td>Adresse 2</td>
  </tr>
</table>


<table>
  <caption>Etudiant BTS SN</caption>
  <tr>
    <th>Nom</th>
    <td>Nom 1</td>
```

```
    <td>Nom 2</td>
  </tr>
  <tr>
    <th>Age</th>
    <td colspan="2">Age 1</td>
  </tr>
  <tr>
    <th>Adresse</th>
    <td>Adresse 1</td>
    <td>Adresse 2</td>
  </tr>
</table>
```

---

7) LES FORMULAIRES :

⇨ **<form> </form>.** C'est la balise principale du formulaire, elle permet d'en indiquer le début et la fin.

```
<form>
  <p>Texte à l'intérieur du formulaire</p>
</form>
```

attribut de la balise <form>

- method :
  - method="get" : c'est une méthode en général assez peu adaptée car elle est limitée à 255 caractères. La particularité vient du fait que les informations seront envoyées dans l'adresse de la page (http://…), mais ce détail ne nous intéresse pas vraiment pour le moment. La plupart du temps, je vous recommande d'utiliser l'autre méthode : post.
  - method="post" : c'est la méthode la plus utilisée pour les formulaires car elle permet d'envoyer un grand nombre d'informations. Les données saisies dans le formulaire ne transitent pas par la barre d'adresse.

- action :

c'est l'adresse de la page ou du programme qui va traiter les informations

- autocomplete: Le navigateur peut remplir automatiquement ou non les champs
  - on ou off

```
<form method="post" action="traitement.php" autocomplete="off">
  <p>Texte à l'intérieur du formulaire</p>
</form>
```

Zones de saisie (entre les balises <form> et </form>) :

⇨ Zone de texte monoligne

<label for="pseudo">Votre pseudo :</label>

```
<input type="text" name="pseudo" id="pseudo" placeholder="Entrer votre texte" size="30" maxlength="10" />
```

- On peut agrandir le champ avec **size.**
- On peut limiter le nombre de caractères que l'on peut saisir avec **maxlength.**
- On peut pré-remplir le champ avec une valeur par défaut à l'aide de **value.**
- On peut donner une indication sur le contenu du champ avec **placeholder.** Cette indication disparaîtra dès que le visiteur aura cliqué à l'intérieur du champ.

⇨ Zone de mot de passe

```
<input type="password" name="pass" id="pass" />
```

⇨ Zone de texte multiligne

```
<textarea name="description" id="description" rows="10" cols="50">
  Texte multilignes
</textarea>
```

⇨ Zone d' e-mail (HTML5)

```
<input type="email" />
```

⇨ Zone de saisie d'url (HTML5)
<input type="url" />

⇨ Zone de saisie de téléphone (HTML5)

```
<input type="tel" pattern="^((\+\d{1,3}(-| )?\(?\d\)?(-| )?\d{1,5})|(\(?\d{2,6}\)?))(-| )?(\d{3,4})(-| )?(\d{4})(( x|ext)\d{1,5}){0,1}$" />
```

⇨ Zone de saisie de nombre entier (HTML5)
<input type="number" step="8" value="0" min="0" max="64" />

⇨ Zone curseur (HTML5)

```
<input type="range" value="15" max="50" min="0" step="5">
```

⇨ Zone de couleur (HTML5)

```
<input type="color" value="#fad345" name="textcolor">
```

⇨ Zone de date (HTML5)

```
<input type="date" max="2012-06-25" min="2011-08-13" name="date">
```

⇨ Zone de recherche (HTML5)

```
<input type="search" placeholder="Entrez un mot-clef" name="search">
```

⇨ Les cases à cocher (checked : coché par défaut)

```
<input type="checkbox" name="choix" checked />
```

⇨ Zones d'options (un choix parmi plusieurs)

```
<input type="radio" name="age" value="moins15" id="moins15" /> <label for="moins15">Moins de 15 ans</label><br />
<input type="radio" name="age" value="medium15-25" id="medium15-25" /> <label for="medium15-25">15-25 ans</label><br />
<input type="radio" name="age" value="medium25-40" id="medium25-40" /> <label for="medium25-40">25-40 ans</label>
```

⇨ Listes déroulantes

```
<select name="pays" id="pays">
    <optgroup label="Europe">
        <option value="france" selected>France</option>
        <option value="espagne">Espagne</option>
        <option value="italie">Italie</option>
        <option value="royaume-uni">Royaume-Uni</option>
    </optgroup>
    <optgroup label="Amérique">
        <option value="canada">Canada</option>
        <option value="etats-unis">Etats-Unis</option>
    </optgroup>
    <optgroup label="Asie">
        <option value="chine">Chine</option>
        <option value="japon">Japon</option>
    </optgroup>
</select>
```

Si le formulaire grossit et comporte beaucoup de champs, il peut être utile de les regrouper au sein de plusieurs balises <fieldset>. Chaque <fieldset> peut contenir une légende avec la balise <legend>.

```
<fieldset>
    <legend>Vos coordonnées</legend> <!-- Titre du fieldset -->

    <label for="nom">Quel est votre nom ?</label>
    <input type="text" name="nom" id="nom" />

    <label for="prenom">Quel est votre prénom ?</label>
    <input type="text" name="prenom" id="prenom" />

    <label for="email">Quel est votre e-mail ?</label>
    <input type="email" name="email" id="email" />
```

</fieldset>

⇨ Rendre un champ obligatoire
<input type="text" name="prenom" id="prenom" required />

Il existe des pseudos classes que l'on peut utiliser pour les éléments <input>.
Ex :

```
input:invalid {
  background-color: #ffdddd;
}

input:valid {
  background-color: #ddffdd;
}

input:required {
  border-color: #800000;
  border-width: 3px;
}
```

⇨ Bouton d'envoi
<input type="submit" value="Envoyer" />

---

8) BALISES AUDIO ET VIDEO EN HTML5 :

Avant l'arrivée de HTML5, il fallait à la place utiliser un plugin, comme Flash.
Il existe plusieurs formats audio et vidéo. Il faut notamment connaître :

- ⇨ pour l'audio : MP3 et Ogg Vorbis (libre) ;
- ⇨ pour la vidéo : H.264, Ogg Theora(libre) et WebM (libre).

Aucun format n'est reconnu par l'ensemble des navigateurs : il faut proposer différentes versions de sa musique ou de sa vidéo pour satisfaire tous les navigateurs.

Insertion d'un fichier audio :

⇨ **<audio src="musique.mp3" controls>**Veuillez mettre à jour votre navigateur ! ou solution flash de secours</**audio>**

On peut compléter la balise des attributs suivants :

- controls : pour ajouter les boutons « Lecture », « Pause » et la barre de défilement. Cela peut sembler indispensable, et vous vous demandez peut-être pourquoi cela n'y figure pas par défaut, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript.
- width : pour modifier la largeur de l'outil de lecture audio.
- loop : la musique sera jouée en boucle.
- autoplay : la musique sera jouée dès le chargement de la page. Évitez d'en abuser, c'est en général irritant d'arriver sur un site qui joue de la musique tout seul !
- preload : indique si la musique peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :
  - auto (par défaut) : le navigateur décide s'il doit précharger toute la musique, uniquement les métadonnées ou rien du tout.
  - metadata : charge uniquement les métadonnées (durée, etc.).
  - none : pas de préchargement. Utile si vous ne voulez pas gaspiller de bande passante sur votre site.

Proposer plusieurs fichiers (le navigateur prend automatiquement le format qu'il reconnaît)

⇨
```
<audio controls>
    <source src="fichier.mp3"></source>
    <source src="fichier.ogg"></source>
</audio>
```

Insertion d'un fichier vidéo :

⇨ **<video src="fichier.webm" controls poster="image.jpg" width="600">** Veuillez mettre à jour votre navigateur ! **</video>**

On peut compléter la balise des attributs suivants :

- poster : image à afficher à la place de la vidéo tant que celle-ci n'est pas lancée. Par défaut, le navigateur prend la première image de la vidéo mais, comme il s'agit souvent d'une image noire ou d'une image peu représentative de la vidéo, je vous conseille d'en créer une ! Vous pouvez tout simplement faire une capture d'écran d'un moment de la vidéo.
- controls : pour ajouter les boutons « Lecture », « Pause » et la barre de défilement. Cela peut sembler indispensable, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript. En ce qui nous concerne, ce sera largement suffisant !
- width : pour modifier la largeur de la vidéo.
- height : pour modifier la hauteur de la vidéo.
- loop : la vidéo sera jouée en boucle.
- autoplay : la vidéo sera jouée dès le chargement de la page. Là encore, évitez d'en abuser, c'est en général irritant d'arriver sur un site qui lance quelque chose tout seul !
- preload : indique si la vidéo peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :
    - auto (par défaut) : le navigateur décide s'il doit précharger toute la vidéo, uniquement les métadonnées ou rien du tout.
    - metadata : charge uniquement les métadonnées (durée, dimensions, etc.).
    - none : pas de préchargement. Utile si vous souhaitez éviter le gaspillage de bande passante sur votre site.

Proposer plusieurs fichiers (le navigateur prend automatiquement le format qu'il reconnaît)

⇨
```
<video controls poster="image.jpg" width="600">
    <source src="fichier.mp4" />
    <source src="fichier.webm" />
    <source src="fichier.ogv" />
</video>
```

III) METTRE EN PLACE LE CSS :

Le CSS (Cascading Style Sheets) mets en forme la page web (couleur du texte, polices, bordures, fond, placer du texte à gauche…)
En CSS, le plus difficile est de savoir cibler le texte dont on veut changer la forme. Pour cibler (on dit « sélectionner ») les éléments de la page à modifier, on utilise **ce qu'on appelle des sélecteurs.**

On peut écrire du langage CSS à trois endroits différents :

- ⇨ dans un fichier **.css** (méthode la plus recommandée)
  Si on écrit le CSS dans un fichier externe (*style.css*), on l'insère ensuite dans l'entête **<head>** de la page html :
  <link rel="stylesheet" href="style.css" /> : c'est elle qui indique que ce fichier HTML est associé à un fichier appelé style.css et chargé de la mise en forme.

- ⇨ dans l'en-tête <head> du fichier HTML ;
  La deuxième méthode consiste à insérer le code CSS directement dans une balise **<style>** à l'intérieur de l'en-tête **<head>.**

```
<style>
    p
    {
       color: blue;
    }
</style>
```

- ⇨ directement dans les balises du fichier HTML via un attribut style (méthode la moins recommandée).
  <p style="color: blue;">texte affcher<p>

1) APPLIQUER UN STYLE :

Schématiquement, une feuille de style CSS ressemble à :

```
balise1
{
   propriete1: valeur1;
   propriete2: valeur2;
   propriete3: valeur3;
}

balise2
{
   propriete1: valeur1;
   propriete2: valeur2;
   propriete3: valeur3;
   propriete4: valeur4;
}
```

Si deux balises doivent avoir la même présentation, une méthode plus rapide consiste à écrire :

```
balise1, balise2
{
   propriete1: valeur1;
   propriete2: valeur2;
   propriete3: valeur3;
   propriete4: valeur4;
}
```

Commentaire dans un fichier CSS : **/*votre commentaire */**

Pour distinguer la mise en forme de plusieurs balises identiques, on utilise l'attribut **class** et **id** et un nom qui sert d'identification.
Le seul différence entre class et id est que l'id ne peut être utilisé qu'une seul fois.
Exemples :

```
<h1 class="titre_principal"> </h1>
<p class="paragraphe"> </p>
<img id="image" />
```

** **
** **

Dans le fichier CSS, on précédera la class par un point et l'id par un dièse.

```
.titre_principal
{
   propriete1: valeur1;
   propriete2: valeur2;
   propriete3: valeur3;
   propriete4: valeur4;
}

#monbloc
{
propriete1: valeur1;
propriete2: valeur2;
}
```

Les sélecteurs avancés :

⇨ * : sélecteur universel (sélectionne toutes les balises)

```
*
{
propriete1: valeur1;
}
```

⇨ A B : une balise contenue dans une autre

```
h3 em
{
propriete1: valeur1;

}
```

⇨ A + B : une balise qui en suit une autre

```
h3 + p
{
propriete1: valeur1;

}
```

Affecte la balise <p> située juste après la balise <h3>

⇨ A[attribut] : une balise qui possède un attribut

```
a[title]
{
propriete1: valeur1;

}
```

Sélectionne tous les liens <a> qui possèdent un attribut title.

⇨ A[attribut="Valeur"] : une balise, un attribut et une valeur exacte

```
a[title="Cliquez ici"]
{
propriete1: valeur1;

}
```

Sélectionne tous les liens <a> qui possèdent un attribut title avec la valeur exacte « Cliquez ici »

⇨ A[attribut*="Valeur"] : une balise, un attribut et une valeur

```
a[title*="ici"]
{
propriete1: valeur1;
}
```

l'attribut doit cette fois contenir dans sa valeur le mot « ici » (peu importe sa position).

Il exister plusieurs autres sélecteurs, voir : http://www.w3.org/Style/css3-selectors-updates/WD-css3-selectors-20010126.fr.html#selectors
http://www.yoyodesign.org/doc/w3c/css2/selector.html

2) LE CSS , L'HERITAGE ET LES PSEUDOS CLASSES :

En CSS, si vous appliquez un style à une balise, toutes les balises qui se trouvent à l'intérieur prendront le même style. Si j'applique une couleur de fond noire et une couleur de texte blanche à la balise <body>, tous mes titres et paragraphes auront eux aussi un arrière-plan de couleur noire et un texte de couleur blanche… C'est ce phénomène qu'on appelle l'héritage : on dit que les balises qui se trouvent à l'intérieur d'une autre balise « héritent » de ses propriétés.

a) L'Héritage par imbrication des Classes ou des Id :

Comme il existe un héritage entre les différentes balises HTML existantes, il est également tout à fait possible d'appliquer cet héritage à vos classes personnalisées ou à vos id.
Pour cela, il suffit dans votre code CSS d'écrire les classes les unes à côté des autres ***séparées par un espace***: la première sera parente de la deuxième, qui sera parente de la troisième, etc... à condition que chacune soit imbriquée dans l'autre au sein de votre code HTML.
Les éléments enfants sont affectés tant que cet élément enfant n'a pas une valeur propre d'affectée pour la propriété.

⇨ .menu li {
propriete1: valeur1;
}
 sera enfant de la classe .menu. Cela va affecter tous les éléments <li> contenus dans la classe .menu et uniquement ceux-là.

⇨ li .menu {
propriete1: valeur1;
}
".menu" sera enfant de la balise <li>. Cela va affecter tous les éléments de classe "menu" contenus dans un

⇨ li, .menu {
propriete1: valeur1;
}
il n'y a plus de notion d'héritage si les éléments sont séparés par une virgule. Cet exemple va signifier ".menu" OU <li>. Cela va affecter tous les éléments de classe "menu" du document, ainsi que toutes les balises <li> du document

⇨ .class1, .class2{
propriete1: valeur1;
}
Propriétés communes aux deux classes (class1 et class2)

⇨ li.menu {
propriete1: valeur1;
}
Ici, cela ne va concerner que les <li> de classe "menu" il s'agit purement et simplement de l'équivalent du "ET" : les deux conditions doivent être réunies pour fonctionner. Il n'y a plus de notion d'héritage.

⇨ #header .callout { } *(espace entre header et .callout)*
Sélectionne tous les éléments avec la classe callout qui sont descendants de l'élément d'id header.

```
<div id= "header" >
    <div class="callout" >
    </div>
</div>
```

⇨ #header.callout { } *(pas d'espace entre header et .callout)*
Sélectionne l'élément qui a l'id header et aussi la class callout.

```
<div id= "header" class="callout" >

</div>
```

⇨ .three.four { color: red; } *(Sélecteur de classe multiple)*

Cible un élément qui a plusieurs classes.
<h1 class="three four">Double Class</h1>

⇨ On peut combiner autant de classes et d'id que nous voulons :

```
.snippet#header.code.red { color: red; }
```

⇨ .class1 > .lien2

```
{
background-color : red;
}
```

(le chevron est sélecteur d'enfant ou de descendant direct. Il ne va donc cibler que les fils, et non tous les descendants, comme le fait " " (l'espace). Dans l'exemple ci-dessous seul les liens **lien 1** et **lien 4** sont concernés.

```
<span class="class1">
    <a href ="#" class = "lien2"> lien 2 </a><br/>

        <div>
            <a href ="#" class = "lien2"> lien 3 </a>
        </div>
    <a href ="#" class = "lien2"> lien 4 </a><br/>
</span>
```

Le chevron sélectionne les premiers éléments enfants (la première descendance)

Exemples:

```
div#id1.class1 div div .class2#id2 {
                color:red;
        }


<div id="id1" class="class1">
        <div>
                <div>
                        <div class="class2" id="id2">
                                eric
                        </div>
                </div>

        </div>
</div>
```

```
span div #id1 div span div.class1 p{
                color:red;
        }

<span>
        <div>
                <div id="id1">
                        <div>
                                <span>
                                        <div class="class1">
                                                <p>eric</p>
                                        </div>
                                </span>
                        </div>
                </div>
        </div>

</span>
```

b) <u>Modifications dynamiques :</u>

Le CSS nous permet aussi de modifier l'apparence des éléments de façon dynamique, c'est-à-dire que des éléments peuvent changer de forme une fois que la page a été chargée.

On pourra changer l'apparence grâce aux **pseudos classes** suivantes :

- ⇨ **:link** (lien par défaut, si aucun événement intervient)
- ⇨ **:visited** (lien déjà visité)
- ⇨ **:hover** (lien au survol)
- ⇨ **:active** (lien actif) => agit au moment du click
- ⇨ **:focus** lors du focus (applique un style lorsque l'élément est sélectionné) ;

Exemples:

⇨ a:hover {
color: red;
}
Colore le texte en rouge lors du survol du lien

⇨ a:focus
{
background-color: #FFCC66;
}

<u>Attention</u> :Vous devez déclarer "a:link et a:visited" avant de déclarer "a:hover" pour un bon fonctionnement

IV) PROPRIETES CSS :

1) FORMATAGE DU TEXTE :

Quelques exemples :

⇨ balise
{
font-family: police1, police2, police3, police4;
}

Le navigateur essaiera d'abord d'utiliser la police1. S'il ne l'a pas, il essaiera la police2. S'il ne l'a pas, il passera à la police3, et ainsi de suite.
En général, on indique en tout dernier serif, ce qui correspond à une police par défaut (qui ne s'applique que si aucune autre police n'a été trouvée).

Avec CSS 3, on peut maintenant utiliser n'importe quelle police :

⇨ aller chercher une police sur http://www.dafont.com/fr/
⇨ Attention tous les formats ne marchent pas sur tous les navigateurs
⇨ @font-face {
font-family: 'MaPolice';
src: url('MaPolice.eot') format('eot'),
url('MaPolice.woff') format('woff'),
url('MaPolice.ttf') format('truetype'),
url('MaPolice.svg') format('svg');
}
h1 /* Utilisation de la police qu'on vient de définir sur les titres */
{
font-family: 'MaPolice', Arial, serif;
}

Résumé propriétés de mise en forme du texte :

Propriété	Valeurs (exemples)	Description
font-family	*police1, police2, police3,* serif, sans-serif, monospace	Nom de police
@font-face	*Nom et source de la police*	Police personnalisée
font-size	1.3em, 16px, 120%...	Taille du texte
font-weight	bold, normal	Gras
font-style	italic, oblique, normal	Italique
text-decoration	underline, overline, line-through, blink, none	Soulignement, ligne au-dessus, barré ou clignotant
font-variant	small-caps, normal	Petites capitales
text-transform	capitalize, lowercase, uppercase	modifier l'aspect des caractères d'un texte (majuscules ou minuscules).
font	-	Super propriété de police. Combine : font-weight, font-style, font-size, font-variant, font-family. (l'ordre n'est pas important)
text-align	left, center, right, justify	Alignement horizontal
vertical-align	baseline, middle, sub, super, top, bottom	Alignement vertical (cellules de tableau ou éléments inline-block uniquement)
line-height	18px, 120%, normal...	Hauteur de ligne
text-indent	25px	Alinéa
white-space	pre, nowrap, normal	Césure
word-wrap	break-word, normal	Césure forcée
text-shadow	5px 5px 2px blue	Ombre de texte

	(horizontale, verticale, fondu, couleur)	

2) LA COULEUR ET LE FOND :

Mode RGB (valeur de 0 à 255)
mode RGBA , on rajoute en plus l'opacité (de 0 à 1)

Mode hexadécimal. (base 16 : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

Base 16	Base 10			
6C5 =	6 x 16^2	+ 12 x 16	+ 5	= 1 733

conversion FF et décimal => FF = $15x16^1 + 15 \times 16^0 = 255$

#FF0000 (les deux première valeur : proportion de rouge, les deux suivantes proportion de vert, les deux suivantes proportion de bleu)
Rem : si on met #FFF => #FFFFFF ; #c36 => #cc3366 (on double)

https://coolors.co/ => générateur de palettes de couleurs

Propriété	Valeurs (exemples)	Description
color	*nom*, rgb(rouge,vert,bleu), rgba(rouge,vert,bleu,transparence), #CF1A20...	Couleur du texte
background-color	*Identique à color*	Couleur de fond
background-image	url('image.png')	Image de fond
background-attachment	fixed, scroll	Fond fixe
background-repeat	repeat-x, repeat-y, no-repeat, repeat	Répétition du fond
background-position	*(x y)*, top, center, bottom, left, right	Position du fond
background	-	Super propriété du fond. Combine : background-image, background-repeat, background-attachment, background-position (l'ordre n'est pas important) . On peut éventuellement mettre deux images
background (Dégradés linéaires)	linear-gradient (direction du dégradé (optionnel),couleur de départ,couleur de fin) linear-gradient (top,red,orange,blue) linear-gradient(45deg,red,orange,blue) Prefixe: Pour firefox : **-moz-** Pour chrome,safari : **-webkit-** Pour opera: **-o-** Pour Internet Explorer: **-ms-**	créer des dégradés en fond de nos éléments HTML.
background (Dégradés radiaux)	radial-gradient(centre du dégradé (optionnel),taille et forme (optionnel),couleur de départ,couleur de fin) radial-gradient(circle,red,orange,blue)	créer des dégradés radiaux

	Prefixe: Pour firefox : **-moz-** Pour chrome,safari : **-webkit-** Pour opera: **-o-** Pour Internet Explorer: **-ms-**	
opacity	0.5	Transparence

3) <u>PROPRIETES DES BOITES :</u>

Une page web peut être vue comme une succession et un empilement de boîtes, qu'on appelle « blocs ». Par exemple , , <p>, <header>, <article>, <nav>…

Propriété	Valeurs (exemples)	Description
width	150px, 80%...	Largeur
height	150px, 80%...	Hauteur
min-width	150px, 80%...	Largeur minimale
max-width	150px, 80%...	Largeur maximale
min-height	150px, 80%...	Hauteur minimale
max-height	150px, 80%...	Hauteur maximale
margin-top	23px	Marge en haut
margin-left	23px	Marge à gauche
margin-right	23px	Marge à droite
margin-bottom	23px	Marge en bas
margin	23px 5px 23px 5px *(haut, droite, bas, gauche)*	Super-propriété de marge. Combine : margin-top, margin-right, margin-bottom, margin-left.
padding-left	23px	Marge intérieure à gauche
padding-right	23px	Marge intérieure à droite
padding-bottom	23px	Marge intérieure en bas
padding-top	23px	Marge intérieure en haut
padding	23px 5px 23px 5px *(haut, droite, bas, gauche)*	Super-propriété de marge intérieure. Combine : padding-top, padding-right, padding-bottom, padding-left.
border-width	3px	Épaisseur de bordure
border-color	*nom*, rgb(rouge,vert,bleu), rgba(rouge,vert,bleu,transparence), #CF1A20...	Couleur de bordure
border-style	solid, dotted, dashed, double, groove, ridge, inset, outset	Type de bordure
border	3px solid black	Super-propriété de bordure. Combine border-width, border-color, border-style. Existe aussi en version border-top, border-right, border-bottom, border-left.
border-radius	10px 5px 10px 5px *(en haut à gauche ,en haut à droite, en bas à droite ,en bas à gauche.)*	Bordure arrondie
box-sizing	content-box: ne prends pas en compte les bordures et les padding (défault)	définit la façon dont la hauteur et la largeur totale d'un élément est calculée

	border-box: prends en compte les bordures dans la largeur et hauteur	
Box-shadow	6px 6px 0px black *(horizontale, verticale, fondu, couleur)*	Ombre de boîte. Si on rajoute inset : l'ombre est à l'intérieur : box-shadow :5px 5px 2px red inset ;

Remarque:
En CSS, la largeur et la hauteur affectées à un élément s'appliquent par défaut à la boîte de contenu (content box) de l'élément. Si l'élément possède une bordure (border) ou du remplissage (padding), celui-ci est ajouté à la largeur et/ou à la hauteur de la boîte affichée à l'écran. Cela signifie qu'il faut ajuster les valeurs de hauteur et de largeur afin qu'elles permettent d'ajouter n'importe quelle bordure ou n'importe quel remplissage qui serait ajouté par la suite.

La propriété **box-sizing** peut être utilisée afin d'ajuster ce comportement :

- **content-box** est la valeur par défaut et correspond au comportement par défaut décrit ci-dessus.
- **border-box** indique au navigateur de prendre en compte la bordure et le remplissage dans la valeur définie pour la largeur et la hauteur.

4) PROPRIETES DE POSITIONNEMENT ET D'AFFICHAGE :

Propriété	Valeurs (exemples)	Description
display	block, inline, inline-block, table, table-cell, none...	Type d'élément (block, inline, inline-block, none…)
visibility	visible, hidden	Visibilité
clip	rect (0px, 60px, 30px, 0px) *rect (haut, droite, bas, gauche)*	Affichage d'une partie de l'élément
overflow	auto, scroll, visible, hidden	Comportement en cas de dépassement
float	left, right, none	Flottant
clear	left, right, both, none	Arrêt d'un flottant
position	relative, absolute, static, fixed	Positionnement
top	20px	Position par rapport au haut
bottom	20px	Position par rapport au bas
left	20px	Position par rapport à la gauche
right	20px	Position par rapport à la droite
z-index	10	Ordre d'affichage en cas de superposition. La plus grande valeur est affichée par-dessus les autres.

a) La propriété CSS display

Par défaut, les éléments HTML vont s'afficher soit sous forme de « bloc », soit « en ligne ». C'est la différence majeure entre les éléments de type block et inline.
La propriété **display** peut prendre de nombreuses valeurs nous permettant de gérer précisément la façon dont chaque élément va être affiché.

b) Display : inline

Lorsque l'on donne la valeur **inline** à la propriété **display**, l'élément ciblé va se comporter comme un élément de **type inline** et donc n'occuper que la largeur qui lui est strictement nécessaire.

Par exemple, en appliquant un **display:inline** à des éléments HTML **p**, ceux-ci vont venir se coller les uns à côté des autres selon la place disponible.

c) Display : block

En attribuant la valeur **block** à la propriété **display**, les éléments ciblés vont se comporter comme des éléments HTML de type block et ainsi prendre toute la largeur disponible.
On peut par exemple appliquer un **display:block** à un élément HTML **span** (qui est par défaut de type inline) afin d'observer la modification du comportement de celui-ci.

d) Display : inline-block

La valeur inline-block va nous permettre d'emprunter certaines propriétés des éléments de type block et certaines propriétés des éléments de type inline. Ainsi, l'élément en soi va être de type inline tandis que ce qu'il contient (« l'intérieur de la boîte ») va être considéré comme étant de type block.

Cela va nous permettre entre autres d'afficher plusieurs éléments côte–à-côte tout en maîtrisant précisément la taille de chacun d'entre eux.

e) Display : none

La valeur **none** va nous permettre tout simplement de ne pas afficher un élément. Cela peut se révéler très pratique dans de nombreux cas, notamment lorsqu'on veut modifier l'affichage de nos pages selon l'appareil utilisé par nos visiteurs (ordinateur de bureau, téléphone portable, tablette, etc.).

f) Display: table:

Les éléments de display **table**, **table-row**, **table-cell** adopte exactement le même rendu visuel qu'un tableau **<table>** avec ses lignes **<tr>** et ses cellules **<td>.** La mise en forme est entièrement réalisée via CSS.

```
<style>

div div div{
display: table-cell;
width:200px;
height:50px;
border:solid;
vertical-align:middle; /* centrer vertical  */
text-align:center; /* centrer horizontal */


}
</style>

<div style="display: table;border-spacing:5px 10px;">
   <div style="display: table-row;">
                <div >
                1er cellule
                </div>
                <div >
                <img src="image.jpg" />
                </div>
   </div>

   <div style="display: table-row;">
                <div >
                3ième cellule
                </div>
                <div >
                4ieme cellule
                </div>
   </div>
</div>
```

g) Positionnement:

On va pouvoir gérer le positionnement de nos éléments HTML en CSS grâce à la propriété CSS **position.**
Grâce à **position**, nous allons pouvoir positionner nos différents éléments HTML de façon absolue ou relativement par rapport à d'autres éléments HTML ou par rapport à leur place d'origine entre autres.
Nous allons utiliser la propriété **position** conjointement aux propriétés **top, right, bottom et left** afin de positionner précisément nos éléments.
Ces quatre propriétés vont pouvoir prendre des valeurs en pixels qui vont indiquer un déplacement d'un élément par rapport à sa position initiale sur un certain bord.
Par défaut, les éléments d'une page ne sont pas positionnés (ils sont en position:**static**).

Position : relative

Un élément positionné grâce à **position:relative** va être repositionné relativement **par rapport à sa position par défaut** dans le code html.
Par exemple, si on positionne un élément HTML de façon relative et qu'on lui ajoute left:50px, l'élément sera déplacé de 50 pixels vers la droite par rapport à sa position par défaut.

1) créer un nouveau référent pour les éléments enfants et descendants **positionnés en absolu**
2) décaler légèrement (de quelques pixels, pas plus) un élément par rapport à sa position normale,

Position : absolute

La valeur absolute de la propriété **position** va nous permettre de positionner un élément de façon **relative** par rapport à son parent le plus proche auquel on a appliqué un positionnement spécifique (relative, fixed ou absolute).
Attention : si l'élément auquel on applique **position:absolute** ne possède pas d'élément parent positionné spécifiquement, celui-ci va se positionner par rapport **à la page entière.**

Position : fixed

Un élément HTML possédant un positionnement « **fixed** » va toujours rester à la même place, même si l'un de vos visiteurs descend ("scroll") dans la page. Un élément positionné en fixed est positionné par rapport **à la fenêtre du navigateur,**
Cette valeur est très utile pour conserver un élément constamment visible, comme un menu ou un sommaire par exemple.

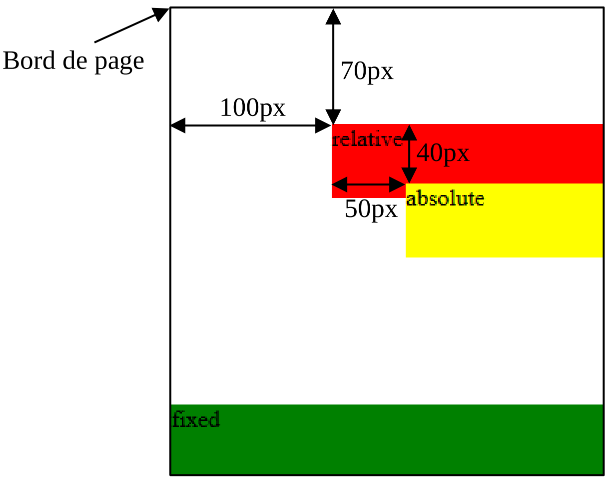
Position : static

La valeur **static** correspond au positionnement par défaut des éléments HTML dans une page.
Tout élément positionné de façon « **static** » ne pourra pas être affecté par les propriétés **top, right, bottom et left.**

Exemple:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>Titre</title>

<style>
body {padding 0;margin: 0}

#relative{
width:100%;
height:50px;
background-color:red;
left:100px;
top:70px;
position:relative;

}
```

```
#absolute{
width:100%;
height:50px;
background-color:yellow;
position:absolute;
   left:50px;
top:40px;
}


#fixed{
width:100%;
height:50px;
background-color:green;
position:fixed;
 bottom: 0;
  right: 0;
}

</style>

   </head>

   <body>
      <div id="relative">
         relative
         <div id="absolute">
            absolute
         </div>
      </div>

      <div id="fixed">
         fixed
      </div>



<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br
/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>

<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br
/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
   </body>
</html>
```



h) Le z-index

La propriété position nous permet de modifier et de casser le flux normal des éléments HTML dans la page en les positionnant où l'on souhaite dans la page.
Parfois, en modifiant la position des éléments ou pas, certains vont pouvoir se chevaucher. Par défaut, l'élément positionné en dernier apparaitra par-dessus les autres.

Cependant, on peut choisir quel élément va apparaître au-dessus de quel autre grâce à la propriété **z-index.**

La propriété z-index va prendre un nombre en valeur : un nombre plus grand positionnera un élément devant un autre ayant un z-index plus petit.

Rem:
Seuls les éléments positionnés peuvent avoir un z-index. Un élément positionné est un élément dont la propriété CSS position a pour valeur **relative, absolute ou fixed.** Par défaut, les éléments d'une page ne sont pas positionnés (ils sont en position:**static**).

i) Le flottement: float et clear:

La propriété float:
La propriété float va nous permettre de faire « flotter » des éléments HTML à gauche ou à droite dans une page web.
On utilisera généralement soit la valeur right (l'élément ciblé flotte à droite), soit left (l'élément ciblé flotte à gauche) avec cette propriété.

La propriété float va faire sortir un élément du flux normal de la page et être placé sur le côté droit ou sur le côté gauche de son conteneur. Il est décalé vers la droite ou vers la gauche jusqu'à ce **qu'il touche le bord de son conteneur *ou un autre élément flottant*.** Il y a trois choses à retenir lorsque l'on utilise cette propriété :

- Les éléments suivants un élément flottant vont se positionner à côté de celui-ci par défaut. Nous allons pouvoir annuler ce comportement grâce à la propriété clear ;
- Les élément positionnés de façon absolue avec position:absolute ignoreront la propriété float (ils ne pourront pas flotter) ;
- Un élément flottera toujours dans les limites (en terme de largeur) de son élément parent conteneur.

La propriété clear:
Elle nous permettre d'empêcher des éléments de venir se positionner aux côtés d'éléments flottants.
Cette propriété peut prendre les valeurs suivantes :

- Left : neutralise l'effet d'un float:left ;
- Right : neutralise l'effet d'un float:right ;
- Both : neutralise l'effet d'un float:left et d'un float:right ;

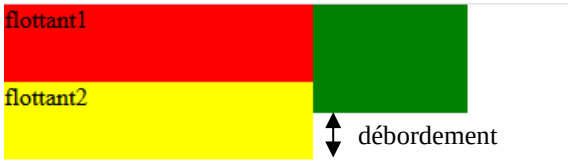

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>Titre</title>

<style>

body {padding 0;margin: 0}

#flottant1{
width:200px;
height:50px;
float:left;
background-color:red;
```

```
}

#flottant2{
width:200px;
height:50px;
background-color:yellow;
float:left;

}

</style>

  </head>
  <body>

    <div id="flottant1">
      flottant1
    </div>

    <div id="flottant2">
        flottant2
    </div>

</html>
```

j) Gestion du débordement (l'overflow)

En utilisant la propriété float, celle-ci va également sortir complètement un élément HTML du flux normal de notre page web. Ainsi, si un élément flottant est plus grand que son élément parent conteneur, celui-ci va déborder de son conteneur verticalement.

Afin d'éviter ce comportement, on peut utiliser la propriété **overflow** qui va nous permettre de cacher (ou pas) ce qui va dépasser.



Cette propriété peut prendre les valeurs suivantes :

- Visible : valeur par défaut (rien ne sera coupé) ;
- Hidden : ce qui dépasse sera coupé ;
- Scroll : coupe de qui dépasse et ajoute une barre de défilement afin d'avoir accès à tout le contenu ;
- Auto : Laisse le navigateur décider du meilleur choix ;

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Titre</title>

<style>

body {padding 0;margin: 0}

#flottant1{
width:200px;
height:50px;
```

```
float:left;
background-color:red;


}

#flottant2{
width:200px;
height:50px;
background-color:yellow;
float:left;

}

#conteneur{
width: 300px;
background-color: green;
height: 70px;
overflow:visible;
}

</style>

  </head>
  <body>

<div id="conteneur">
    <div id="flottant1">
      flottant1
    </div>

    <div id="flottant2">
        flottant2
    </div>
  </div>
</html>
```

k) le Flexbox CSS:

Parfois les propriétés display et ses différentes valeurs block, inline et inline-block ainsi que sur les propriétés float et position amène à des résultats inattendus lors de changement dynamiques.
On introduit la nouvelle valeur **flex** pour la propriété **display** ainsi que la propriété **flex** (et toutes ses sous propriétés).

Le flexbox est aujourd'hui l'outil le plus puissant et simple pour créer des structures responsives et flexibles.
En effet, le Flexbox va nous permettre de gérer précisément :

- La direction des éléments (en colonne ou en ligne, avec un retour à la ligne ou non) ;
- L'alignement des éléments selon un axe principal et secondaire ainsi que leur répartition ;
- L'ordre des éléments ;
- La place prise par les éléments en fonction de l'espace disponible.

Le Flexbox va fonctionner sur le principe de conteneurs et d'éléments contenus.
On commence par définir un **élément conteneur** (qui sera souvent un div) dans lequel nous allons placer les différents éléments que l'on souhaite positionner dans la page et aligner les uns par rapport aux autres et qui formeront un groupe.

Il faut appliquer un **display : flex** (type block) ou éventuellement un **display : inline-flex** (type inline) à notre conteneur.

Grâce au **display : flex** du conteneur, les éléments contenus deviennent flexibles (éléments de type flex-item)

Gérer la direction des flex-items avec le Flexbox:
La propriété flex-direction (pour le conteneur) peut prendre les valeurs suivantes :

- Row (valeur par défaut) : les éléments sont organisés horizontalement ;

- Row-reverse : les éléments sont organisés horizontalement mais dans l'ordre inverse ;
- Column : les éléments sont organisés verticalement ;
- Column-reverse : les éléments sont organisés verticalement mais dans l'ordre inverse.

La propriété flex-wrap qui va nous permettre de définir si le contenu doit être affiché sur une seule ligne (ou colonne) ou sur plusieurs.
La propriété flex-wrap (pour le conteneur) peut prendre les valeurs suivantes :

- Nowrap : les éléments ne peuvent pas passer à la ligne ;
- Wrap : les éléments peuvent passer à la ligne dans leur ordre naturel de déclaration ; si jamais le conteneur devient trop petit. Ainsi, grâce à cette propriété, nous sommes certains que notre conteneur englobera toujours ses éléments flex-items et que les éléments conserveront la largeur qu'on leur a définie.
- Wrap-reverse : les éléments peuvent passer à la ligne dans l'ordre inverse de leur déclaration.


```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>Titre</title>

<style>

body {padding 0;margin: 0}

#conteneur{
background-color:blue;
width:700px;
display : flex;
flex-direction:row;
flex-wrap:wrap;

}

#element1
{
background-color:red;
padding:20px;
width:250px;
}

#element2
{
background-color:yellow;
padding:20px;
width:250px;
}

#element3
{
background-color:green;
padding:20px;
width:250px;
}
```

```
</style>

  </head>
  <body>

<div id="conteneur">
    <div id="element1">
      Element 1
    </div>

    <div id="element2">
        Element 2
    </div>

    <div id="element3">
        Element 3
    </div>
  </div>
</html>
```

Gérer l'alignement avec le Flexbox:

Nous allons pouvoir définir l'organisation des éléments selon l'axe principal des éléments (horizontal ou vertical) avec la propriété **justify-content**. L'organisation des éléments selon l'axe secondaire va ensuite pouvoir être gérée avec les propriétés **align-items, align-self** et **align-content.**

Si on décide d'organiser nos éléments de **manière horizontale**, alors l'axe principal des éléments sera **horizontal** tandis que l'axe secondaire sera **vertical** et inversement.

La propriété justify-content (axe principal) va pouvoir prendre les valeurs suivantes :

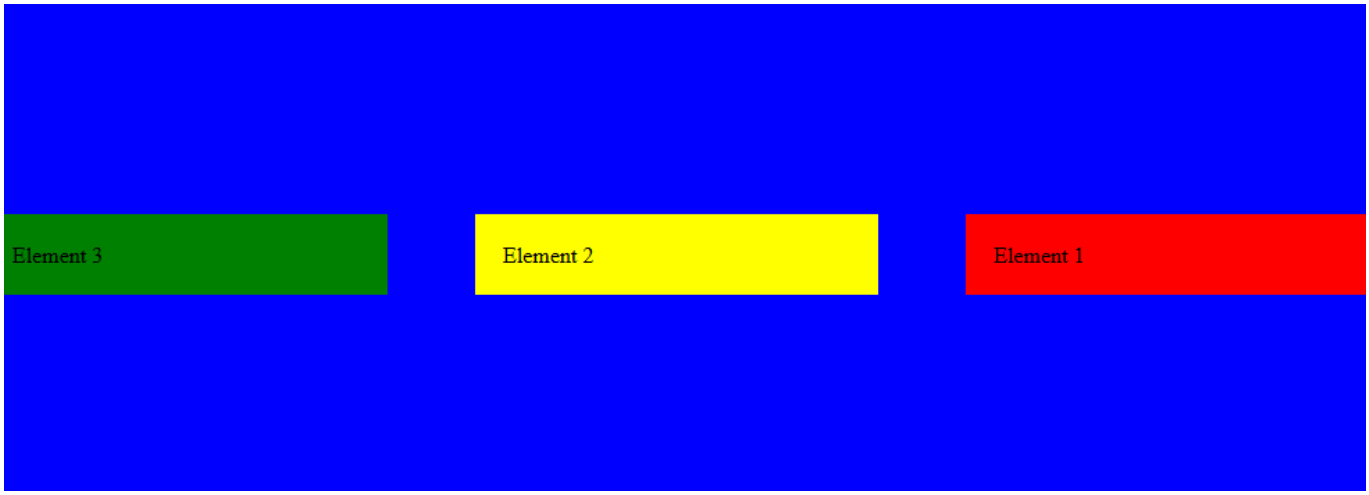
- Flex-start (défaut) : les éléments s'alignent en début de ligne ;
- Flex-end : les éléments s'alignent en fin de ligne ;
- Center : les éléments s'alignent au centre ;
- Space-between : les éléments sont distribués régulièrement sur la ligne avec le premier élément en tout début de ligne et le dernier en toute fin ;
- Space-around : les éléments sont distribués régulièrement sur la ligne et s'espacent les uns des autres de manière identique (notez que l'espace entre le premier / dernier élément et le bord est deux fois moins grands qu'entre deux éléments) ;
- Space-evenly : les éléments sont distribués régulièrement sur la ligne et s'espacent les uns des autres et des bords de manière identique.


```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>Titre</title>

<style>

body {padding 0;margin: 0}

#conteneur{
background-color:blue;
display : flex;
```

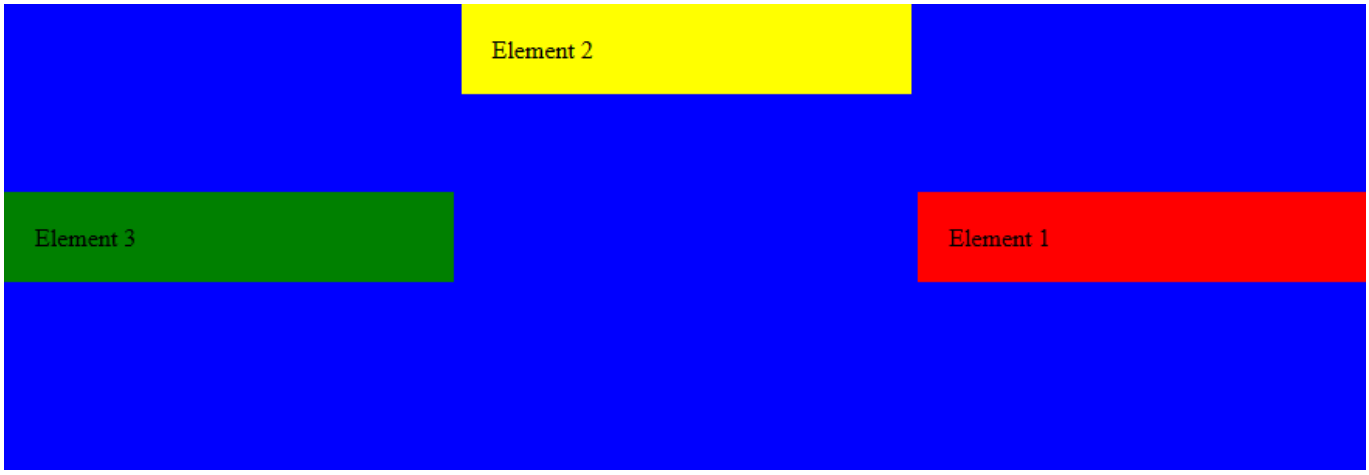
```
flex-direction:row-reverse;
flex-wrap:wrap;
justify-content:Space-between;
}

#element1
{
background-color:red;
padding:20px;
width:250px;
}

#element2
{
background-color:yellow;
padding:20px;
width:250px;
}

#element3
{
background-color:green;
padding:20px;
width:250px;
}


</style>

  </head>
  <body>

 <div id="conteneur">
     <div id="element1">
       Element 1
     </div>

     <div id="element2">
          Element 2
     </div>

     <div id="element3">
          Element 3
     </div>
  </div>
</html>
```

La propriété align-items: (axe secondaire): va nous permettre de définir le comportement des éléments le long de leur axe secondaire. Elle possède donc un rôle similaire à la propriété justify-content mais pour l'axe secondaire.
Cette propriété va également **s'appliquer au conteneur** et va pouvoir prendre les valeurs suivantes :

- Stretch (défaut) : les éléments sont étirés sur tout l'axe ;
- Flex-start : les éléments sont alignés au début de l'axe ;
- Flex-end : les éléments sont alignés à la fin de l'axe ;
- Center : les éléments sont centrés par rapport à l'axe ;
- Baseline : les éléments sont alignés de telle sorte à ce que leurs lignes de base soient alignées les unes par rapport aux autres.



On peut organiser un élément en particulier grâce à la propriété **align-self**. Notez que cette propriété sera prioritaire sur **align-items.**
align-self peut prendre les mêmes valeurs que **align-items**. Elle va s'appliquer à un élément flex-item en particulier et non pas au conteneur.

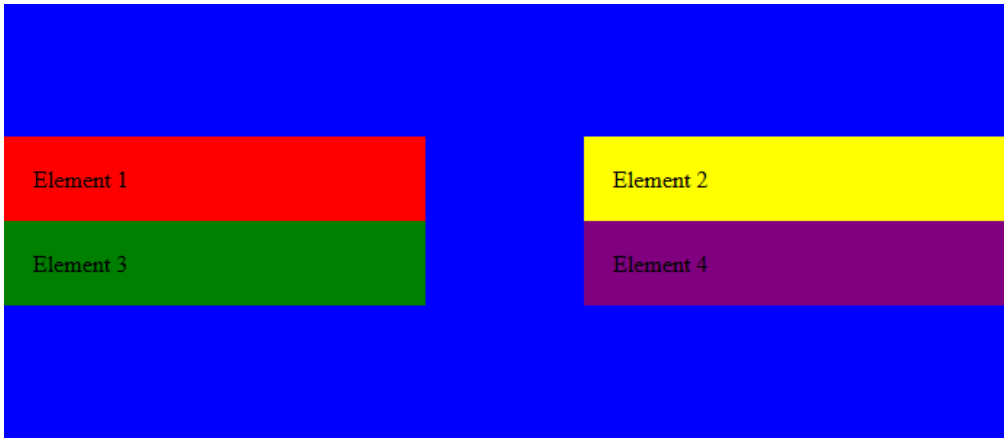


Dans le cas où nos **flex-items s'affichent sur plusieurs lignes**, le Flexbox va également nous permettre de gérer l'organisation de chaque ligne avec la propriété **align-content** par rapport à l'axe secondaire.

Notez que cette propriété n'aura aucun effet si notre Flexbox ne contient qu'une ligne.

La propriété **align-content** va pouvoir prendre les valeurs suivantes :

- Stretch (par défaut) : les éléments s'étendent pour prendre toute la place ;
- Flex-start : les lignes se collent au début du conteneur ;
- Flex-end : les lignes se collent en fin du conteneur ;
- Center : les lignes se centrent dans le conteneur ;
- Space-between : les lignes sont distribuées régulièrement avec la première ligne au début du conteneur et la dernière à la fin de celui-ci ;
- Space-around : les lignes sont distribuées régulièrement avec le même espace entre chaque ligne.



5) <u>LES TRANSITIONS EN CSS:</u>

Les transitions vont nous permettre de modifier la valeur d'une propriété CSS de façon fluide, dans le temps, créant ainsi une transition entre les différentes valeurs de notre propriété. On va par exemple pouvoir changer progressivement la couleur de nos textes.

La propriété CSS **transition** est une écriture short hand regroupant en fait quatre propriétés :

- **Transition-delay** : indique quand doit commencer la transition ;
- **Transition-duration** : indique la durée de la transition ;
- **Transition-property** : définit la propriété à laquelle la transition doit s'appliquer ;
  all: *(Valeur par défaut)* Toutes les propriétés animables seront animées
- **Transition-timing-function** : définit la courbe de vitesse de notre transition.
  - Ease : La transition commence lentement puis accélère au milieu pour finir lentement
  - Ease-in : la transition commence lentement ;
  - Ease-out : la transition se termine lentement ;
  - Ease-in-out : la transition commence et se finit lentement ;
  - Linear : transition linéaire.

Parmi ces quatre propriétés, seules les valeurs de **transition-duration et transition-property** vont être obligatoires à préciser pour utiliser la propriété transition.

Ces deux valeurs vont donc correspondre à la durée de la transition en secondes (« s ») ainsi qu'au nom de la propriété CSS à laquelle on souhaite appliquer la transition.

Nous utiliserons souvent les transitions avec les pseudo classes et particulièrement avec **:hover.**

On utilise également les préfixes suivant les navigateurs:

- Pour firefox : **-moz-**
- Pour chrome : **-webkit-**
- Pour opera,safari : **-o-**
- Pour Internet Explorer: **-ms-**

```
<!DOCTYPE html>
<html>
   <head>
      <title>Transitions !</title>
      <meta charset= "utf-8">

      <style>
      .div-un{
   width: 200px;
   height: 200px;
   background-color: #06D;
   border: 5px solid purple;
   -webkit-transition: width 2s, background-color 5s;
   -moz-transition: width 2s, background-color 5s;
   -o-transition: width 2s, background-color 5s;
   transition: width 2s, background-color 5s;
      }
```

```
.div-un:hover{
   width: 400px;
   background-color: red;
}
      </style>

   </head>

   <body>

      <div class="div-un">
         <p>Un premier paragraphe</p>
         <p>Un deuxième paragraphe</p>
         <p>Un troisième paragraphe</p>
      </div>
   </body>
</html>
```

```
<!DOCTYPE html>
<html>
   <head>
      <title>Transitions !</title>
      <meta charset= "utf-8">

<style>
.div-un{
   width: 200px;
   height: 200px;
   background-color: #06D;
   border: 5px solid purple;
   -webkit-transition: 3s width 2s ease, background-color 5s ease;
   -moz-transition: 3s width 2s ease, background-color 5s ease;
   -o-transition: 3s width 2s ease, background-color 5s ease;
   transition: 3s width 2s ease, background-color 5s ease;
}

.div-un:hover{
   width: 400px;
   background-color: red;
}
</style>

   </head>

   <body>

      <div class="div-un">
         <p>Un premier paragraphe</p>
         <p>Un deuxième paragraphe</p>
         <p>Un troisième paragraphe</p>
      </div>
   </body>
</html>
```

6) PROPRIETE TRANSFORM:

La propriété **transform** permet de modifier l'espace de coordonnées utilisé pour la mise en forme visuelle. Grâce à cette propriété, il est possible de **translater** les éléments, de les **tourner**, d'appliquer des **homothéties**, de les **distordre** pour en changer la perspective.
On peut appliquer plusieurs transformations.
Les transformations qui sont composées entre elles sont appliquées dans l'ordre, de gauche à droite.

Pour pouvoir appliquer des transformations, nous avons besoin de savoir quel est le point d'origine (d'ancrage) de la transformation. La propriété **transform-origin** définit ce point d'origine.
La valeur initiale de cette propriété est le centre de l'élément: **transform-origin: 50% 50%;**
Il est possible de changer cette valeur en utilisant un mot-clef de position (top, right, bottom, left) ou une valeur chiffrée dont l'unité peut varier (px, %, etc.)
transform-origin: top left ; transform-origin: 50px 50px;

scale (1,2) : zoom horizontal de 1 et zoom vertical de 2
transform : rotate(45deg) ; rotation de 45°
transform : skew(45deg) ; oblique de 45°
transform : translate(20px, 100px) ; translation de 20px vers la droite et 100px vers le bas (on peut mettre des valeurs négatives. (vers la gauche et vers le haut)

liste de fonctions prédéfinies:

- none : aucune transformation, valeur par défaut.
- matrix() : pour appliquer une matrice de transformation à un élément.
- translate(x,y) : pour déplacer horizontalement et verticalement un élément.
- translateX() : pour déplacer horizontalement un élément.
- translateY() : pour déplacer verticalement un élément.
- translateZ() : pour déplacer en profondeur un élément.
- translate3d(x,y,z) : pour déplacer sur les trois axes de perspective un élément.
- scale(x,y) : pour redimensionner horizontalement et verticalement un élément.
- scaleX() : pour redimensionner horizontalement un élément.
- scaleY() : pour redimensionner verticalement un élément.
- scaleZ() : pour redimensionner un élément sur sa profondeur.
- scale3d(x,y,z) : pour redimensionner un élément sur les trois axes de perspective.
- rotate() : pour faire une rotation à un élément en degré: rotate(45deg)
- rotateX() : pour faire une rotation à un élément sur l'axe de perspective X.
- rotateY() : pour faire une rotation à un élément sur l'axe de perspective Y.
- rotateZ() : pour faire une rotation à un élément sur l'axe de perspective Z.
- rotate3d() : pour faire une rotation à un élément sur les trois axes de perspective.
- skew() : pour incliner horizontalement et verticalement un élément en degré: skew(45deg)
- skewX() : pour incliner horizontalement un élément.
- skewY() : pour incliner verticalement un élément.

```
<!DOCTYPE html>
<html>
   <head>
      <title>Transitions !</title>
      <meta charset= "utf-8">

<style>
.div1{
   width: 200px;
   height: 200px;
   background-color: #06D;
   border: 5px solid purple;

   transition:transform 5s;
}

.div2{
   width: 200px;
   height: 200px;
   background-color: #06D;
```

```
   border: 5px solid purple;

   transition:transform 5s;
   position:relative;
   left:300px;
}


.div1:hover{
   transform:translate(300px,0px) rotate(45deg)   ;
}

.div2:hover{
   transform-origin: top  left  ;
   transform: rotate(360deg)   ;
}


</style>

   </head>

   <body>

      <div class="div1">
         <p>Un premier paragraphe</p>
         <p>Un deuxième paragraphe</p>
         <p>Un troisième paragraphe</p>
      </div>

      <div class="div2">
         <p>Un premier paragraphe</p>
         <p>Un deuxième paragraphe</p>
         <p>Un troisième paragraphe</p>
      </div>

   </body>
</html>
```

7) LES ANIMATIONS EN CSS:

Les animations en CSS vont nous permettre de changer le style d'un élément HTML.
Contrairement aux **transitions**, on va pouvoir grâce aux **animations** modifier le style d'un élément HTML sans changement d'état de celui-ci (par exemple avec hover => voir transition) et autant de fois que voulu.

Les animations sont une fonctionnalité relativement récente du CSS. Ainsi, la compatibilité avec d'anciennes versions de navigateurs n'est pas parfaite et nous allons devoir utiliser des préfixes vendeurs

Pour firefox : **-moz-**
Pour chrome,safari : **-webkit-**
Pour opera : **-o-**
Pour Internet Explorer: **-ms-**

Pour créer une animation en CSS, nous allons utiliser la propriété **animation** ainsi que la règle CSS **@keyframes.**

On appelle **@keyframes** une « règle » en CSS, qui va nous permettre de modifier progressivement le style d'un élément.
La propriété CSS **animation** est en fait la notation short-hand des propriétés relatives aux animations suivantes :

- **animation-name** : nom de l'animation, qu'on réutilisera dans la déclaration du @keyframes ;
- **animation-duration** : durée de l'animation, en secondes ;
- **animation-timing-function** : courbe de vitesse de l'animation ;
- **animation-delay** : délai de l'animation ;
- **animation-iteration-count** : nombre de fois que l'animation doit être jouée ;
- **animation-direction** : spécifie si l'animation doit se jouer à l'envers ou selon des cycles alternés, c'est-à-dire en changeant de sens à chaque fois.

Parmi toutes ces propriétés, seules les valeurs relatives aux deux premières (**animation-name** et **animation-duration**) doivent obligatoirement être précisées dans la propriété CSS animation.
Une fois notre animation terminée un élément reprend son style d'origine.

```
<!DOCTYPE html>
<html>
   <head>
      <title>Animations CSS !</title>
      <meta charset= "utf-8">

   <style>
   div {
   width: 100px;
   height: 100px;
   background-color: blue;
   -webkit-animation-name: chgt-couleur; /* Chrome, Safari, Opera */
   -webkit-animation-duration: 3s;
    -webkit-animation-iteration-count: infinite;
 -webkit-animation-direction: alternate;

   -moz-animation-name: chgt-couleur; /* Mozilla */
   -moz-animation-duration: 3s;
    -moz-animation-iteration-count: infinite;
 -moz-animation-direction: alternate;

   animation-name: chgt-couleur;
   animation-duration: 3s;
    animation-iteration-count: infinite;
 animation-direction: alternate;
}

/*Pour Chrome, Safari et Opéra*/
@-webkit-keyframes chgt-couleur {
   0% {background-color: blue;}
   100% {background-color: orange;}
}

/*Pour Mozilla*/
```

```
@-moz-keyframes chgt-couleur{
   0% {background-color: blue;}
   100% {background-color: orange;}
}

/*Syntaxe standard*/
@keyframes chgt-couleur {
   0% {background-color: blue;}
   100% {background-color: orange;}
}
   </style>

   </head>

   <body>

      <div></div>
   </body>
</html>
```

8) PROPRIETES DES LISTES :

Propriété	Valeurs (exemples)	Description
list-style-type	disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none	Type de liste
list-style-position	inside, outside	Position en retrait
list-style-image	url('puce.png')	Puce personnalisée
list-style	-	Super-propriété de liste. Combine list-style-type, list-style-position, list-style-image.

9) PROPRIETES DES TABLEAUX :

Propriété	Valeurs (exemples)	Description
border-collapse	collapse, separate	Fusion des bordures
empty-cells	hide, show	Affichage des cellules vides
caption-side	bottom, top	Position du titre du tableau

V) MISE EN PAGE ADAPTATIVE :

Le « viewport » correspond à la taille de la fenêtre web de vos visiteurs.
On va demander à ce que nos pages web s'adaptent à la taille de la fenêtre de chacun de nos visiteurs, afin que notre contenu la remplisse.
<meta name="viewport" content="width=device-width, initial-scale=1.0" />

En plus, grâce aux media queries nous pourrons créer un design qui s'adapte automatiquement à l'écran de chaque visiteur !
Les media queries sont donc des règles qui indiquent quand on doit appliquer des propriétés CSS.
Il y a deux façons de les utiliser :

- ⇨ en chargeant une feuille de style .css différente en fonction de la règle (ex : « Si la résolution est inférieure à 1280px de large, charge le fichier petite_resolution.css ») ;
  <link rel="stylesheet" media="screen and (max-width: 1280px)" href="petite_resolution.css" />

- ⇨ en écrivant la règle directement dans le fichier .css habituel (ex : « Si la résolution est inférieure à 1280px de large, charge les propriétés CSS ci-dessous »).

```
@media screen and (max-width: 1280px)
{
   propriete1: valeur1;
   propriete2: valeur2;
}

/* Sur tous types d'écran, quand la largeur de la fenêtre est comprise entre 1024px et 1280px */
@media all and (min-width: 1024px) and (max-width: 1280px)
{
   propriete1: valeur1;
   propriete2: valeur2;
}
```

Règles disponibles:

- ⇨ color : gestion de la couleur (en bits/pixel).
- ⇨ height : hauteur de la zone d'affichage (fenêtre).
- ⇨ width : largeur de la zone d'affichage (fenêtre).
- ⇨ device-height : hauteur du périphérique.
- ⇨ device-width : largeur du périphérique.
- ⇨ orientation : orientation du périphérique (portrait ou paysage).
- ⇨ media : type d'écran de sortie. Quelques-unes des valeurs possibles :
  - screen : écran « classique » ;
  - handheld : périphérique mobile ;
  - print : impression ;
  - tv : télévision ;
  - projection : projecteur ;
  - all : tous les types d'écran.

Voici donc les points de rupture souvent utilisés pour un affichage « normal
Jusqu'à 479px : smartphone en portrait
De 480px à 959px : smartphone en paysage, tablette en portrait et petite tablette en paysage
De 960px à 1280px : tablette en paysage, écran d'ordinateur de taille petite et moyenne
1281px et au delà : grand écran d'ordinateur *(21" et + en plein écran*)