

Formation web IM@GE 2019-2020

Introduction au web

| | |
|---|----|
| Fonctionnement d'un site internet | 4 |
| Principe client/serveur | 4 |
| Principe de Front End et Back End | 4 |
| Les gestionnaires de code source..... | 5 |
| Le front-end..... | 7 |
| HTML/CSS et rôle du navigateur | 7 |
| HTML | 8 |
| <i>Un langage à balise</i> | 8 |
| Structure générale d'une page HTML | 8 |
| Les commentaires | 9 |
| Manipulation de texte | 9 |
| Les titres | 9 |
| Les paragraphes..... | 10 |
| Accentuation sur l'importance d'un mot ou d'une expression..... | 11 |
| Listes et tableau..... | 13 |
| Les listes..... | 13 |
| Les tableaux..... | 14 |
| Les petits tableaux..... | 14 |
| Les grands tableaux..... | 15 |
| Les liens | 18 |
| Les liens vers une page d'un autre site | 18 |
| Les liens vers une autre page de notre site..... | 20 |
| Les ancrés | 21 |
| Les médias | 23 |
| Les images | 23 |
| Les formats d'images..... | 23 |
| L'insertion d'une image..... | 23 |
| L'image cliquable..... | 24 |
| Le son..... | 24 |
| Les formats de son | 24 |
| L'insertion du son | 24 |
| La vidéo..... | 26 |
| Les figures..... | 26 |

| | |
|---|----|
| Les formulaires | 27 |
| Mettre en place un formulaire | 27 |
| Les champs de saisie..... | 28 |
| Le champ de texte | 28 |
| Le champ de recherche | 29 |
| Le champ d'email | 30 |
| Le champ de mot de passe | 31 |
| Le champ d'URL..... | 31 |
| Le champ de nombre..... | 31 |
| Le champ de numéro de téléphone | 32 |
| Le curseur | 32 |
| La palette de couleur..... | 33 |
| Le formulaire à choix unique..... | 34 |
| Le formulaire à choix multiples | 35 |
| La liste déroulante..... | 35 |
| La zone de texte multiligne | 37 |
| Le regroupement des différents champs | 37 |
| Les conteneurs | 39 |
| Les différents types de conteneurs | 39 |
| L'entête..... | 39 |
| Le pied de page..... | 39 |
| La section..... | 40 |
| L'article | 41 |
| Les informations annexes..... | 42 |
| La navigation | 42 |
| La division du contenu..... | 43 |
| Le modèle des boîtes..... | 44 |
| Rappel des bonnes pratiques en HTML..... | 45 |
| CSS | 46 |
| La syntaxe de base du CSS..... | 46 |
| Trois manières d'écrire du code CSS..... | 46 |
| L'écriture directement dans une balise HTML | 46 |
| L'écriture dans l'entête du fichier HTML..... | 47 |
| La création d'une feuille de style | 48 |
| Sélecteur et style | 49 |
| Les balises HTML | 49 |

| | |
|---|----|
| Les identifiants | 50 |
| Les classes..... | 50 |
| Les balises universelles..... | 51 |
| Les sélecteurs avancés | 52 |
| Les commentaires | 53 |
| La couleur | 53 |
| Les notations | 53 |
| Désignation d'une couleur par son nom | 53 |
| La notation hexadécimale | 54 |
| La notation RGB..... | 54 |
| Ajouter un fond | 55 |
| La couleur de fond..... | 55 |
| L'image de fond | 55 |
| L'opacité de notre fond | 56 |
| La mise en forme du texte..... | 57 |
| La taille | 57 |
| Les effets stylisés sur du texte..... | 57 |
| Mettre en gras..... | 57 |
| La mise en italique..... | 58 |
| La décoration du texte | 58 |
| La police..... | 58 |
| L'alignement..... | 59 |
| Le positionnement flottant | 59 |
| Bordure et ombre..... | 59 |
| Les tableaux suite et fin..... | 59 |

Fonctionnement d'un site internet

Principe client/serveur

Tout le monde a déjà utilisé un site internet, mais derrière l'utilisation facile et intuitive d'une page web, comment tout cela fonctionne-t-il ?

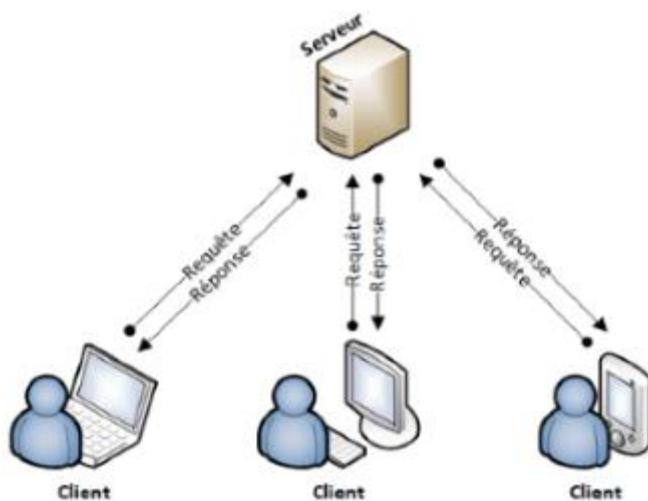
Pour répondre à cette question il faut comprendre le principe de client/serveur.

Lorsque vous consultez un site avec votre machine (ordinateur, téléphone, tablette, etc), vous accédez à des fichiers qui ne sont pas sur votre ordinateur mais sur un serveur qui est bien souvent à l'autre bout du monde. Ces fichiers contiennent le code qui permet de faire fonctionner le site tel que vous le voyez, lui donner son aspect et gérer toutes ses fonctionnalités. Le serveur contient également la base de données.

Mais c'est quoi un serveur au juste ?

Pour faire simple, c'est un ordinateur très puissant, qui est allumé continuellement et qui stocke des ressources accessibles sur internet (les sites internet entre autres).

On dit que votre machine est le **client**, et que le site internet est placé côté **serveur**. Ainsi, le client et le serveur vont communiquer pour réaliser des actions sur votre écran (affichage d'une page stockée sur le serveur, chargement de données stockées sur le serveur, etc). Pour réaliser cette communication, il faut qu'ils se comprennent, qu'ils parlent la même « langue ». Cette langue, c'est ce qu'on appelle le « protocole réseau ».



Vous trouverez des explications beaucoup plus détaillées au sujet des protocoles de communication dans le cours de cyber-sécurité et réseau, ce n'est pas l'objet de ce cours.

Principe de Front End et Back End

Maintenant que nous savons que notre machine communique avec un serveur, nous aimeraisons comprendre quels sont ces fameux fichiers qui sont stockés là-bas. Un peu de patience nous y venons ! D'abord, sachez qu'il existe de nombreux métiers qui gravitent autour de la création d'un site internet, en voici quelques-uns :

- Développeur front-end
- UI designer
- UX designer
- Web designer
- Développeur back-end
- Architecte réseau
- Administrateur de bases de données
- Développeur full-stack
- ...

Ces métiers se comptent par dizaines, certains sont spécialisés dans un domaine précis, d'autres sont plus polyvalents.

Très bien mais quel est le rapport avec tout ce que l'on a pu dire avant ? Et bien il faut pouvoir faire travailler tout ce beau monde en même temps !

Pour cela, on va diviser la création de notre site en deux catégories : le **front-end** et le **back-end**.

Le développement front-end concerne tout ce qui attire à l'apparence, au design de votre site internet ! C'est donc tout ce qui sera affiché à l'écran. Si votre site n'est pas joli, il faudra taper sur les doigts du développeur front-end et des designers !

Si l'on prend l'exemple d'une maison, le front-end, c'est le décorateur, le maçon, le charpentier et le jardinier qui s'en occupent !

Le back-end est la partie complémentaire au front-end, il s'agit de la partie immergée de l'iceberg ! C'est tout ce que l'on ne voit pas au premier coup d'œil mais qui est néanmoins essentiel au fonctionnement de votre site.

En effet, comment afficher des données si ces dernières sont sur le serveur ?

Il faut les charger et c'est le rôle du back-end. Le back-end permet aussi entre autres de réaliser des connexions à un compte utilisateur, de modifier la base de données, etc.

Si on reprend l'analogie de notre maison, le back-end c'est le plombier qui raccorde l'eau au bâtiment ou l'électricien qui réalise le câblage électrique.

Cette division du code en deux parties permet de bien séparer le visuel du fonctionnel. Ainsi, si l'on veut faire une refonte graphique de notre site, il suffira de modifier le front-end sans que le back-end n'en soit altéré.

Remarque : Si vous entendez parler de « full stack », ce terme regroupe le front-end et le back-end (ex : développeur full stack).

Les gestionnaires de code source

Cette séparation du code est pratique pour structurer notre projet et cela facilite un peu le travail en équipe. Mais en pratique, ce sont souvent des équipes de dizaines d'employés qui travaillent sur un même projet. Si tout le monde modifie les mêmes fichiers en même temps, cela crée des conflits de fichier et tout peut-être endommagé.

Comment faire alors pour faire travailler simultanément plusieurs personnes sur un même projet ?

C'est là qu'interviennent les gestionnaires de version. Cette partie du cours ne concerne pas à proprement parlé le développement web mais sera utile quel que soit le projet informatique que vous comptez faire. Mieux vaut prendre les bonnes habitudes dès le début !

Dans ce cas, qu'est-ce qu'un gestionnaire de version ?

PARTIE DU COURS SUR GIT A FAIRE DES QUE LE COURS SUR LE FRONT END EST TERMINE

Le front-end

HTML/CSS et rôle du navigateur

Le front-end, rappelez-vous, est la mise en forme de notre site. Pour cela, on dispose de deux principaux langages informatique, le HTML et le CSS.

Le HTML (*HyperText Markup Language*), c'est le langage qui donne sens à votre page. C'est lui qui définit les éléments de la page, qui dit que tel élément est un titre, tel autre est un paragraphe, que telle partie est un entête ou tel autre un pied de page, etc. Le HTML est interprété par le navigateur web avant d'afficher votre page. Si votre code HTML est bien réalisé, le navigateur va facilement comprendre comment votre code est structuré et le référencement de votre site internet sera amélioré. Le référencement, vous savez, c'est le fait que votre site internet soit dans les premiers résultats lors d'une recherche dans un moteur de recherche ou qu'au contraire il soit plusieurs pages plus loin. N'oubliez jamais que le référencement c'est très important, on fait un site internet pour qu'il soit consulté ! De plus, si votre code HTML est propre, il vous sera plus facile de le modifier par la suite.

Le CSS (*Cascading Style Sheets*) quant à lui est le langage qui permet de faire la mise en page. Ainsi, nous allons pouvoir désigner des éléments du code HTML et modifier leur positionnement, leur taille leur couleur et bien d'autres ! Depuis quelques années, le CSS permet d'adapter la mise en page de notre site en fonction de la taille d'écran. Il s'agit d'un nouvel enjeu devenu très important avec l'essor des tablettes et des téléphones portables. Le fait qu'un site s'adapte aux différentes tailles d'écran est désigné par le terme de site **responsive** ou **implémentant du responsive design**.

Cependant, ce code HTML et CSS n'est pas suffisant tel quel pour pouvoir visualiser un site internet. Pour ce faire, ces fichiers doivent être interprétés par un navigateur web. Des navigateurs web, il en existe une multitude dont en voici les principaux : Internet Explorer, Mozilla Firefox, Google Chrome, Safari, Microsoft Edge ou encore Opéra. Les fonctionnalités de ces derniers sont très similaires, cependant on s'assurera que le rendu de nos pages web est correct sur plusieurs navigateurs car on peut parfois avoir de mauvaises surprises.

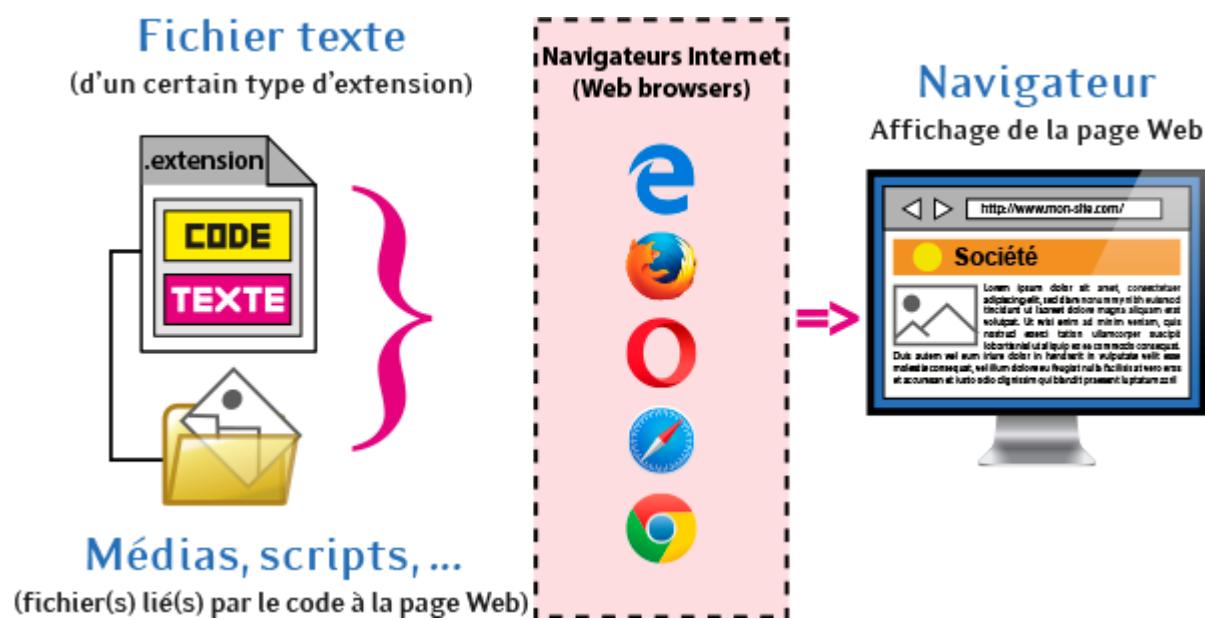


Figure 1: Rôle du navigateur web (Image de imedias.pro)

HTML

Un langage à balise

Le HTML est un langage à balise, la plupart de ses balises s'utilisent avec une balise ouvrante, un contenu puis une balise fermante :

```
<balise ouvrante>Contenu de la balise</balise fermante>
```

Exemple : Le paragraphe

```
<p>Ceci est un paragraphe</p>
```

Certaines balises sont auto-fermantes, aussi appelées **balises orphelines**, elles n'ont pas de contenu :

```
<balise />
```

Exemple : Le retour à la ligne

```
<br />
```

Ensuite, les balises ont des attributs qui permettent de donner des informations à leur sujet.

```
<balise ouvrante attribut1="valeur1", attribut2="valeur2">Contenu de la  
balise</balise fermante>
```

Exemple : Une image se nommant *mon_image.jpg*

```

```

Structure générale d'une page HTML

Avant toute chose, sachez qu'une page HTML a toujours la même structure de base :

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

Ne vous inquiétez pas, ça à l'air compliqué comme ça mais il n'y a rien de difficile. Avec le temps, écrire ce code de base devient même un automatisme !

La première ligne : `<!DOCTYPE html>`, indique au navigateur web que le document que nous sommes en train d'écrire est de type HTML. Facile non ?

Ensuite les balises `<html></html>` délimitent le début et la fin du code html.

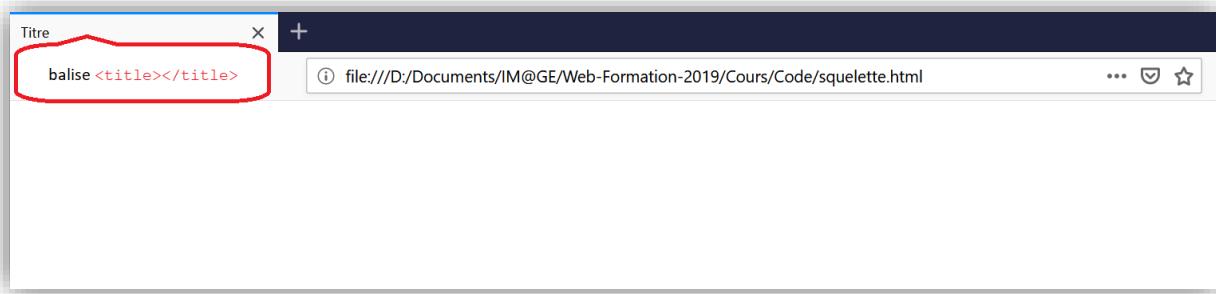
Au sein de ces deux balises on a deux autres balises, la première est la suivante : `<head></head>`.

Il s'agit de l'entête du code html. L'entête contient une ou plusieurs balises `<meta />` et une balise `<title></title>`.

La balise `<title></title>` c'est le titre de votre page qui s'affiche au niveau de l'onglet dans le navigateur web. Les balises `<meta />` quant à elles contiennent des **métadonnées**, il s'agit en fait de données utilisées par les navigateurs et les moteurs de recherches. Certaines sont utiles pour le référencement et d'autres sont devenues obsolètes. Celle que l'on mettra dans tous les cas est : `<meta charset="utf-8" />`. Cela permet de donner le type d'encodage des caractères dans notre fichier html, le navigateur pourra alors reconnaître les caractères un peu spéciaux tels que les accents et les apostrophes par exemple.

Enfin viennent les balises `<body></body>`, elles délimitent le corps du code html. C'est à l'intérieur de ces balises que l'on va pouvoir définir les éléments de notre page.

On obtient alors le résultat suivant :



Les commentaires

En html, on peut écrire des **commentaires** comme suit : `<!-- Contenu du commentaire -->`.

Les commentaires sont souvent négligés, quel que soit le langage utilisé, que ce soit par manque de temps ou par flemme 😊. Cependant, les commentaires permettent d'expliquer certains bouts de votre code, ils sont utiles pour expliciter certains passages un peu compliqués à comprendre à première vue et à faciliter la lecture et la compréhension par un tiers ou par vous-même. Commentez votre code ! Vous me remercieriez quand vous le relirez dans deux ans.

On est d'accord, le commentaire en html n'est pas forcément primordial car le langage est très intuitif et proche du langage naturel. Pour d'autres langages il est bien plus important alors prenons les bonnes habitudes dès le départ !

Manipulation de texte

Les titres

Bien ! Nous pouvons maintenant entrer dans le vif du sujet ! Nous allons enfin pouvoir commencer à écrire notre code html à proprement parler. Souvenez-vous, c'est au sein des balises `<html></html>` que l'on va écrire ce dernier.

Lorsque nous commençons notre page web, la première chose à faire est souvent d'écrire un titre (pas le titre dans l'onglet hein ne confondons pas avec la balise `<title></title>` vue précédemment). Cependant, tous les titres de notre page n'auront pas la même importance. En effet, il faut pouvoir différencier nos titres par niveau (titre général, sous-titre, titre encore plus petit, etc). Pour ce faire, html nous laisse six balises à notre disposition qui sont les suivantes :

```
<h1>Ce titre est méga important !</h1>
<h2>Ce titre est super important !</h2>
<h3>Ce titre est important !</h3>
<h4>Ce titre est moyennement important !</h4>
<h5>Ce titre n'est pas vraiment important !</h5>
<h6>Ce titre n'est pas du tout important !</h6>
```

Le titre `<h1>` étant le plus important et `<h6>` le moins important.

Voici par exemple un aperçu des six différents types de titre.

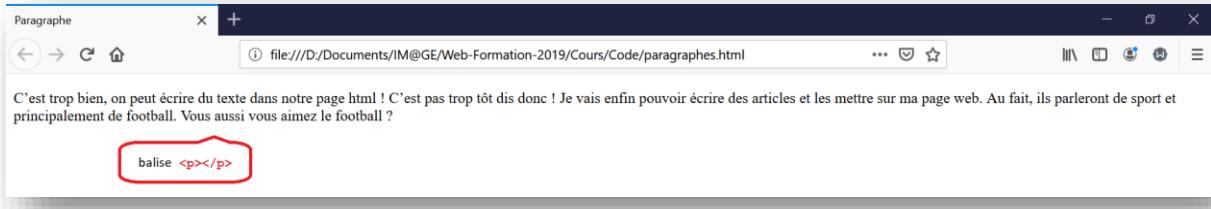


Les paragraphes

Mettre des titres c'est bien beau mais on aimerait maintenant écrire un bloc de texte. En html, cela consiste à créer des paragraphes :

```
<p>C'est trop bien, on peut écrire du texte dans notre page html ! C'est
pas trop tôt dis donc !
Je vais enfin pouvoir écrire des articles et les mettre sur ma page web. Au
fait, ils parleront de sports et principalement de football. Vous aussi
vous aimez le football ?</p>
```

On obtient alors :



Mince ! J'ai mis un retour à la ligne mais lorsque la page s'affiche à l'écran tout reste sur la même ligne ! Au secouuurs je ne comprends pas !

Pas de panique c'est normal ! En html, un paragraphe est un bloc de texte continu. Appuyer sur la touche « entrée » au beau milieu du code n'y changera rien. Mais il y a bien une solution, à vrai dire il y en a même deux. Les deux sont équivalentes, le choix va varier en fonction de la manière dont vous pensez les choses.

En effet, si vous souhaitez retourner à la ligne c'est sûrement parce que votre paragraphe est fini et que vous vous apprêtez tout simplement à en commencer un autre. A ce moment il suffit de créer deux paragraphes au lieu d'un, on a alors un saut de ligne :

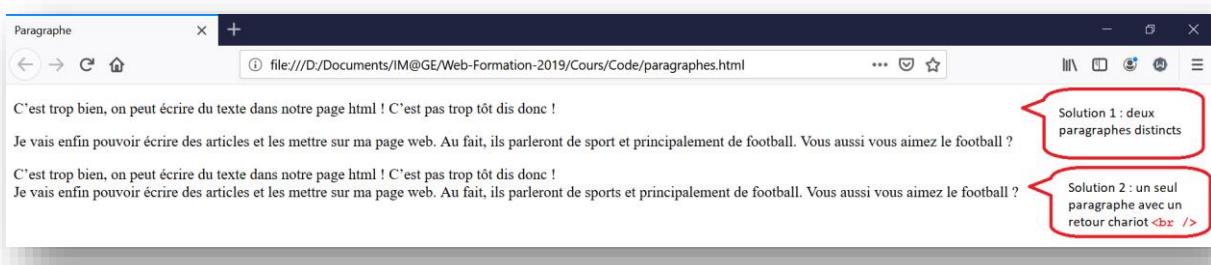
<p>C'est trop bien, on peut écrire du texte dans notre page html ! C'est pas trop tôt dis donc !</p>
<p>Je vais enfin pouvoir écrire des articles et les mettre sur ma page web. Au fait, ils parleront de sports et principalement de football. Vous aussi vous aimez le football ?</p>

Dans le second cas, on veut rester dans le même paragraphe, il s'agit réellement d'un retour à la ligne. La solution est de mettre la balise qui représente le retour à la ligne en html. Nous l'avions vu en début de cours rappelez-vous, il s'agit de la balise orpheline
 :

<p>C'est trop bien, on peut écrire du texte dans notre page html ! C'est pas trop tôt dis donc !

Je vais enfin pouvoir écrire des articles et les mettre sur ma page web. Au fait, ils parleront de sports et principalement de football. Vous aussi vous aimez le football ?</p>

On a donc :



Accentuation sur l'importance d'un mot ou d'une expression

Dans notre texte, on va sûrement vouloir accentuer certains mots pour mieux les mettre en valeur car ce sont eux qui dégagent les idées importantes. Prenons pour exemple le premier paragraphe de la page d'Elon Musk sur Wikipedia :

`<p>Elon Reeve Musk (prononciation : /'i:.lon 'mʌsk/), né le 28 juin 1971 à Pretoria, est un entrepreneur, chef d'entreprise et ingénieur d'origine sud-africaine naturalisé canadien en 1988 puis américain en 2002. Il est actuellement installé à Los Angeles. Il est le PDG et directeur de la technologie de la société SpaceX mais également DG, directeur architecture produit de la société Tesla, et ancien président du conseil d'administration de SolarCity et de Tesla. Il est également le fondateur de The Boring Company, une société de construction de tunnels, et de Neuralink, une société de neurotechnologie.</p>`

Dans ce paragraphe, ce que l'on trouve important c'est le nom complet d'Elon Musk et les différentes entreprises qu'il a pu diriger.

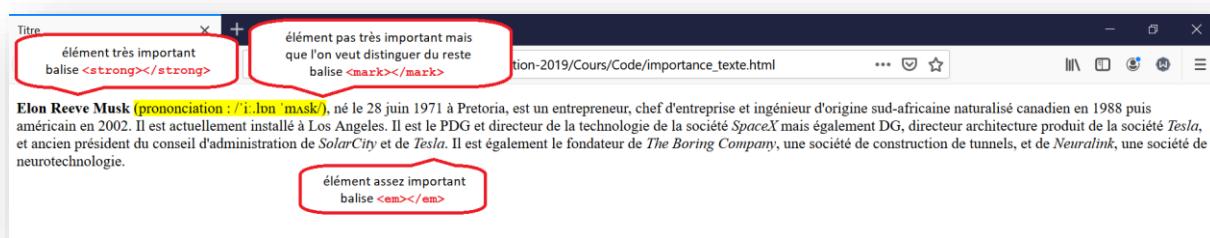
Il y a trois types de mise en valeur possible qui ont des degrés de force différents.

- On veut signifier qu'une expression est très importante : ``
- On veut signifier qu'une expression est assez importante : ``
- L'expression n'est pas spécialement importante, on veut simplement la mettre en valeur pour la distinguer du reste : `<mark></mark>`

Ici, on va fortement accentuer le nom d'Elon Musk, ensuite on considère que les noms des entreprises sont assez importants. Enfin, la prononciation n'est pas spécialement importante mais on aimeraît tout de même la distinguer du reste pour qu'elle soit facilement remarquable. D'où le code suivant :

`<p>Elon Reeve Musk <mark>(prononciation : /'i:.lon 'mʌsk/)</mark>, né le 28 juin 1971 à Pretoria, est un entrepreneur, chef d'entreprise et ingénieur d'origine sud-africaine naturalisé canadien en 1988 puis américain en 2002. Il est actuellement installé à Los Angeles. Il est le PDG et directeur de la technologie de la société SpaceX mais également DG, directeur architecture produit de la société Tesla, et ancien président du conseil d'administration de SolarCity et de Tesla. Il est également le fondateur de The Boring Company, une société de construction de tunnels, et de Neuralink, une société de neurotechnologie.</p>`

On obtient finalement une page avec les différents mots mis en valeur :



Par défaut, vous pouvez voir que le navigateur met le contenu des balises `` en gras, le contenu des balises `` en italique et enfin surligne en jaune le contenu de `<mark></mark>`. En effet, le navigateur comprend qu'il s'agit de mot dont on veut accentuer l'impact et change donc l'apparence par défaut. Cependant, n'oublions pas que le HTML est un langage qui permet de donner la structure de notre page, on utilisera donc ces balises **SEULEMENT** pour accentuer un mot et non pas pour le mettre en gras ou en italique ! La mise en forme est gérée par le CSS et non par le HTML. On verra d'ailleurs plus tard qu'il est facilement possible de changer la mise en forme de ces balises en CSS.

Listes et tableau

Nous savons désormais comment écrire des paragraphes. Cependant, il nous est parfois nécessaire d'utiliser des listes ou des tableaux pour synthétiser nos propos. Ces derniers ont également d'autres avantages, on utilise par exemple les listes en les combinant avec des liens pour créer des menus.

Les listes

Il existe deux types de listes :

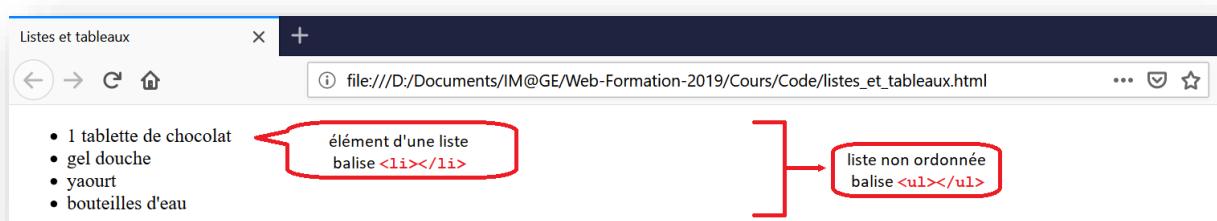
- Les listes à puce : les éléments ne sont pas ordonnés entre eux.
- Les listes ordonnées : les éléments sont numérotés suivant un certain ordre.

Voici un exemple de liste à puce qui représente notre liste de course:

```
<ul>
    <li>1 tablette de chocolat</li>
    <li>gel douche</li>
    <li>yaoourt</li>
    <li>bouteilles d'eau</li>
</ul>
```

`` définit le début et la fin de notre liste non ordonnée (`ul` pour *unordered list*). Ensuite, pour chaque élément de la liste, on le rajoute au sein des balises `` (`li` pour *list item*).

Le résultat est le suivant :



Concernant les listes non ordonnées, voici la syntaxe :

```
<ol>
    <li>Mettez vos pates dans l'eau chaude</li>
    <li>Faites chauffer jusqu'à ébullition</li>
    <li>Laissez cuire vos pâtes entre 9 et 15 minutes</li>
    <li>Salez avec modération et dégustez !</li>
```

```
</ol>
```

La recette de cuisine s'affiche comme suit :



Les tableaux

Bon, les tableaux c'est un tout petit peu plus compliqué que ce que l'on a pu voir à présent. Cela fait intervenir le CSS, ici nous nous intéresserons seulement à la partie html des tableaux.

Il y a deux manières de faire un tableau, la première étant la version simplifiée de la seconde. On utilisera la première solution pour les petits tableaux car on ne veut pas surcharger inutilement de petits tableaux. Vous comprendrez mieux pourquoi la deuxième solution peut être utile dans certains cas.

Les petits tableaux

La structure d'un tableau est la suivante :

```
<table>
  <caption>Effectif des ZZ1 à l'ISIMA</caption>
  <tr>
    <th>Nom</th>
    <th>Prénom</th>
    <th>Groupe</th>
  </tr>
  <tr>
    <td>Marchand</td>
    <td>Pierre</td>
    <td>3</td>
  </tr>
  <tr>
    <td>Moulin</td>
    <td>Marie</td>
    <td>1</td>
  </tr>
</table>
```

Comme vous pouvez le deviner, les balises `<table></table>` délimitent le tableau en lui-même. A l'intérieur de ces balises vont donc se placer les différents éléments qui composent un tableau.

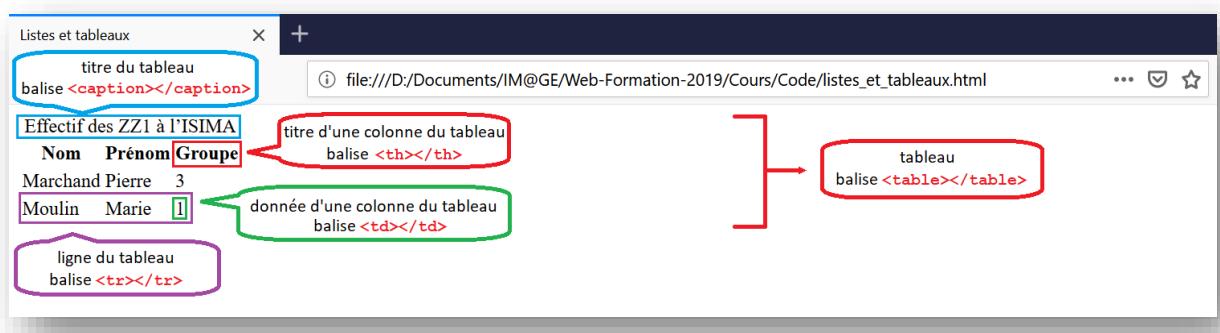
Les balises `<caption></caption>` permettent d'insérer un titre au tableau.

Ensuite viennent les balises `<tr></tr>` (`tr` pour *table row*), elles définissent une ligne de notre tableau. Il y aura donc autant de balises `<tr></tr>` que notre tableau contiendra de lignes. Au sein des balises `<tr></tr>` se trouvent donc les colonnes du tableau.

Le premier `<tr></tr>` contient généralement les titres des différentes colonnes avec les balises `<th></th>` (`th` pour *table header*). On n'est en aucun cas obligé de mettre des titres à nos colonnes, mais c'est quand même mieux pour faciliter la compréhension générale du tableau.

Les balises `<td></td>` (`td` pour *table data*) quant à elles définissent les données que l'on veut mettre sous forme de tableau. Dans le tableau ci-dessous, il s'agit du nom d'une personne ou encore de son prénom, à moins que cela ne soit son groupe.

Le résultat affiché à l'écran :



Mais un tableau ça a des bordures, pourquoi elles n'apparaissent pas ?

Les bordures sont considérées comme de la mise en page, c'est pour cela qu'il faut utiliser du CSS pour les faire apparaître. Ne vous inquiétez pas, on refera un point sur les tableaux plus loin dans le cours, lorsque l'on se sera un peu plus familiarisé avec le CSS.

METTRE UN LIEN VERS LA PARTIE DU COURS CORRESPONDANTE

Les grands tableaux

Considérons désormais un tableau plus exhaustif généré par le code suivant :

```
<table>
  <caption>Effectif des ZZ1 à l'ISIMA</caption>
  <tr>
    <th>Nom</th>
    <th>Prénom</th>
    <th>Groupe</th>
  </tr>
  <tr>
    <td>Marchand</td>
    <td>Pierre</td>
```

```

<td>3</td>
</tr>
<tr>
    <td>Moulin</td>
    <td>Marie</td>
    <td>1</td>
</tr>
<tr>
    <td>Mine</td>
    <td>Hugo</td>
    <td>2</td>
</tr>
<tr>
    <td>Ara</td>
    <td>Frank</td>
    <td>1</td>
</tr>
<tr>
    <td>Ponte</td>
    <td>Valentin</td>
    <td>1</td>
</tr>
<tr>
    <td>Ollier</td>
    <td>Bastien</td>
    <td>3</td>
</tr>
</table>
```

Pour clarifier son code, on peut vouloir rajouter certaines balises qui délimitent les différentes zones de notre tableau. Cela peut être utile lorsque l'on a des très gros tableaux.

On va utiliser les trois balises suivantes :

- **<thead></thead>** : zone de l'entête du tableau
- **<tfoot></tfoot>** : zone du pied du tableau
- **<tbody></tbody>** : zone du corps du tableau

Remarque : il faut agencer les balises dans cet ordre précis. C'est un peu contre-intuitif mais c'est le navigateur qui remettra les éléments dans le bon ordre lors de l'affichage du tableau à l'écran.

```





```

```
<tr>
    <th>Nom</th>
    <th>Prénom</th>
    <th>Groupe</th>
</thead>

<tfoot>
    <th>Nom</th>
    <th>Prénom</th>
    <th>Groupe</th>
</tfoot>

<tbody>
    </tr>
    <tr>
        <td>Marchand</td>
        <td>Pierre</td>
        <td>3</td>
    </tr>
    <tr>
        <td>Moulin</td>
        <td>Marie</td>
        <td>1</td>
    </tr>
    <tr>
        <td>Mine</td>
        <td>Hugo</td>
        <td>2</td>
    </tr>
    <tr>
        <td>Ara</td>
        <td>Frank</td>
        <td>1</td>
    </tr>
    <tr>
        <td>Ponte</td>
        <td>Valentin</td>
        <td>1</td>
    </tr>
    <tr>
        <td>Ollier</td>
```

```

<td>Bastien</td>
<td>3</td>
</tr>
</tbody>
</table>

```

Aperçu du rendu visuel :

| Nom | Prénom | Groupe |
|----------|----------|--------|
| Marchand | Pierre | 3 |
| Moulin | Marie | 1 |
| Mine | Hugo | 2 |
| Ara | Frank | 1 |
| Ponte | Valentin | 1 |
| Ollier | Bastien | 3 |

Remarque : Ces balises ne sont absolument pas obligatoires et ne changent rien à l'aperçu final. Elles permettent seulement d'améliorer la lisibilité de votre code.

Les liens

Bon on a maintenant une page avec du texte et de jolies listes, c'est cool ! Mais, comme vous le savez, l'intérêt d'internet est de pouvoir naviguer entre des pages.

Oui c'est vrai ça ! Comment lier des pages ?

Pour cela, on va utiliser ce qu'on appelle des liens. Il existe trois types de liens :

- Les liens vers une page d'un autre site
- Les liens vers une autre page de notre propre site, typiquement, on les utilise pour faire des sommaires
- Les liens qui permettent de se déplacer dans la page aussi appelés **ancre**s

Les liens vers une page d'un autre site

Reprendons notre paragraphe sur Elon Musk. Rendons à César ce qui appartient à César, nous allons donc mettre un lien vers l'article dont nous avons honteusement copié le début.

Un lien est représenté par les balises `<a>`. Nous allons ajouter à la fin de notre paragraphe le code suivant :

```

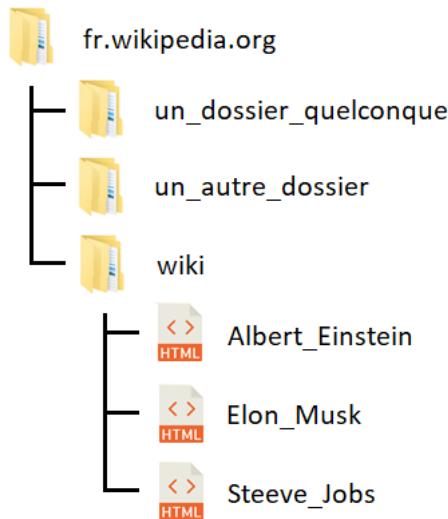
<a href="https://fr.wikipedia.org/wiki/Elon_Musk">L'article Wikipedia
complet ici</a>

```

Comme vous pouvez le constater, cette balise a un attribut `href` qui permet de référencer l'URL (l'adresse) de la page sur laquelle on veut aller que l'on appelle **URL destination**.

Un petit mot sur l'URL tout de même : https://fr.wikipedia.org/wiki/Elon_Musk. Ce type d'URL est une adresse absolue vers la page destination puisque l'adresse commence à la racine de l'arborescence jusqu'à la fin, chaque fils de l'arbre étant rajouté derrière un « / ».

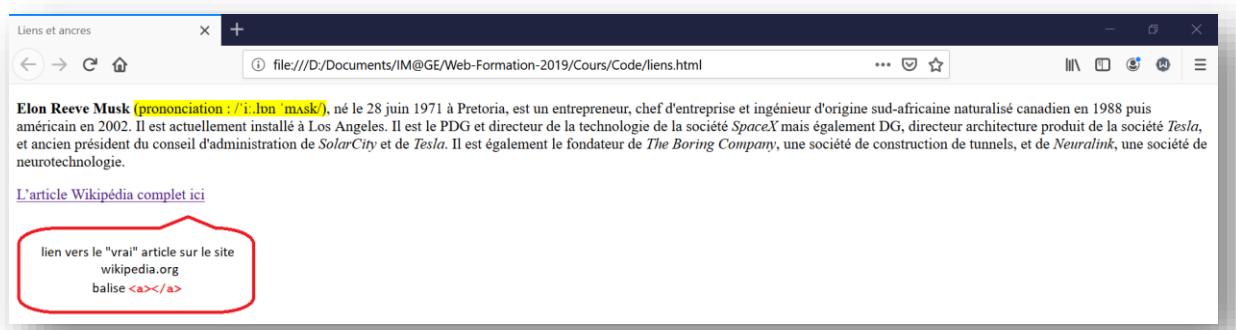
Prenons l'exemple suivant, imaginons que le dossier fr.wikipedia.org contiennent trois dossiers « un dossier quelconque », « un autre dossier » et « wiki » et que le dossier « wiki » contient trois fichiers html qui concernent Albert Einstein, Elon Musk ainsi que Steve Jobs.



Pour aller sur la page web concernant Musk, il faut donc aller dans le dossier « fr.wikipedia.org » puis « wiki » puis aller sur la page d'Elon Musk.

On obtient alors le chemin suivant : /fr.wikipedia.org/wiki/Elon_Musk.html. Ce type de chemin est appelé chemin absolu car il part de la racine de l'arborescence jusqu'à la destination. Le lien que nous avons créé précédemment vers la page wikipedia d'Elon Musk utilise une URL composée d'un chemin absolu, on l'appelle donc un **lien absolu**.

On obtient donc un lien qui est souligné et colorié en bleu par défaut par les navigateurs :



Lorsque l'on clique sur le lien on arrive bien sur l'article en question :

Les liens vers une autre page de notre site

Maintenant, nous aimerais créer notre propre article ! Pour cela nous créons un nouveau fichier html dans le même dossier que le premier dont voici le résultat :

J'ai très légèrement modifié notre premier fichier en y ajoutant le début de notre magnifique article :

Maintenant ajoutons notre nouveau lien. Il est possible de faire un lien absolu comme précédemment :

```
<a href="file:///D:/Documents/IM@GE/Web-Formation-
```

```
2019/Cours/Code/Liens/mon_article.html">Mon article complet ici</a>
```

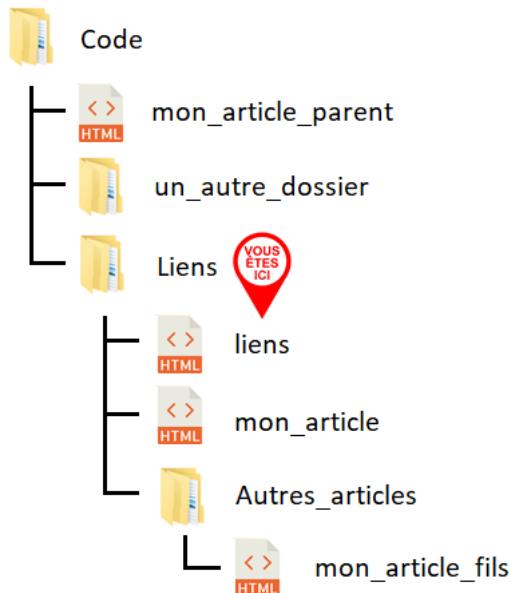
Mais c'est quand même assez barbare, n'est-ce pas ? De plus, si on rajoute un dossier quelque part dans l'arborescence, cela risque de casser tous nos liens et il faudra tout refaire. Imaginez si vous en avez cinquante !

On va donc préférer un autre type de lien, le **lien relatif** : `Mon article complet ici`

C'est beaucoup plus simple, comment as-tu fais cela ?

De la même manière qu'un lien absolu utilise un chemin absolu, un lien relatif utilise un **chemin relatif**.

Imaginons l'arborescence suivante :



Nous souhaitons ajouter notre lien dans le fichier *liens.html*. Notre nouvel article s'intitule *mon_article.html*. Le principe d'une adresse relative est de redéfinir la racine de l'arbre que nous souhaitons parcourir. Pour cela, nous considérons que la nouvelle racine est le dossier contenant notre fichier dans lequel on veut rajouter le lien. Notre nouvelle racine est donc le dossier *Liens*. Hors, comme *mon_article.html* et *liens.html* sont dans le même dossier, l'adresse relative est donc *mon_article.html*.

Si nous souhaitons désormais faire un lien vers une page html qui n'est pas dans le même dossier que notre fichier de base nous avons deux cas de figure :

- Le fichier est dans un dossier père : pour cela on utilise « .. » pour remonter dans l'arborescence. Imaginons que l'on veuille créer un lien vers *mon_article_parent.html* dans le fichier *liens.html*, le chemin relatif sera alors : [.. /mon_article_parent.html](#).
- Le fichier est dans un dossier fils : pour cela, on redescend dans l'arborescence comme d'habitude, le seul changement est la racine qui est redéfinie : [Autres_articles /mon_article_fils.html](#).

Attention ! Il ne faut pas mettre de « / » au tout début du chemin comme pour le chemin absolu. En effet, le premier « / » représente la racine de votre ordinateur, cela n'a donc aucun sens pour un chemin relatif puisque le but même du chemin relatif est de redéfinir cette racine.

Les ancrés

Le dernier type de liens permet de se déplacer au sein de la même page (aller plus bas dans la page ou revenir plus haut). C'est ce que l'on appelle une **ancré**. Ce mécanisme est très utile lorsque nous avons de très grandes pages avec beaucoup de contenu. Pour réaliser cela, on doit faire un lien vers un élément de notre propre page.

Faire un lien vers un élément de notre propre page ? Comment est-ce possible ?

Pour ce faire, on doit pouvoir indiquer vers quel élément se rediriger. Pour identifier cet élément, quoi de plus naturel que d'utiliser un identifiant ! On crée un identifiant avec l'attribut id="mon_id".

Pour que cela ait un peu plus d'intérêt, j'ai pris la permission de rajouter quelques articles, ainsi qu'un petit sommaire, j'espère que vous ne m'en voudrez pas, le résultat final est le suivant :

The screenshot shows a web browser window with the title "Liens et ancrages". The address bar displays the URL "file:///D:/Documents/IM@GE/Web-Formation-2019/Cours/Code/Liens/liens.html#elon_musk". The main content area contains a section titled "Sommaire" with a bulleted list:

- [Elon Musk](#)
- [Mon super article](#)
- [Steve Jobs](#)
- [Jeff Bezos](#)

Below this is a section titled "Mes supers articles" containing:

Elon Musk

Elon Reeve Musk ([pronunciation : /i.lən 'mʌsk/](#)), né le 28 juin 1971 à Pretoria, est un entrepreneur, chef d'entreprise et ingénieur d'origine sud-africaine naturalisé canadien en 1988 puis américain en 2002. Il est actuellement installé à Los Angeles. Il est le PDG et directeur de la technologie de la société SpaceX mais également DG, directeur architecture produit de la société Tesla, et ancien président du conseil d'administration de SolarCity et de Tesla. Il est également le fondateur de The Boring Company, une société de construction de tunnels, et de Neuralink, une société de neurotechnologie.

[L'article Wikipedia complet ici](#)

Youhou mon premier article perso !

Voici mon premier article, je suis vraiment content de vous le montrer ! Je vais enfin pour écrire moi même et arrêter de pomper sur wikipedia car je vais enfin savoir comment lier mes pages entre elles.

[Mon article complet ici](#)

Steve Jobs

J'ai donc ensuite rajouté les identifiants sur chaque titre de mes articles :

```
<h2 id="elon_musk">Elon Musk</h2>
<!-- Du code -->
<h2 id="mon_article">Youhou mon premier article perso !</h2>
<!-- Du code -->
<h2 id="steve_jobs">Steve Jobs</h2>
<!-- Du code -->
<h2 id="jeff_bezos">Jeff Bezos</h2>
<!-- Du code -->
```

Pour réaliser le sommaire, j'ai écrit le code ci-dessous :

```
<ul>
  <li><a href="#elon_musk">Elon Musk</a></li>
  <li><a href="#mon_article">Mon super article</a></li>
  <li><a href="#steve_jobs">Steve Jobs</a></li>
  <li><a href="#jeff_bezos">Jeff Bezos</a></li>
</ul>
```

Tout dans le code ci-dessus ou presque doit vous être familier. J'ai seulement fait un menu en combinant une liste non ordonnée et des liens. La seule chose un peu étonnante qui doit vous frapper à l'esprit est le caractère « # » au niveau de la référence du lien. **Le « # » est le caractère qui permet**

de faire référence à l'identifiant d'un élément HTML. Souvenez-vous en, on en reparlera au niveau du CSS.

Lorsque l'on clique sur le lien de Jeff Bezos, on voit bien que la page scrolle vers le bas jusqu'à l'élément en question.

Récapitulons, pour faire une ancre il faut :

- Ajouter un identifiant unique à chaque élément que l'on veut cibler avec une ancre.
- Ajouter un lien par élément que l'on veut cibler en mettant le caractère « # » suivi de l'identifier de cet élément.

Les médias

On commence à savoir pas mal de chose désormais, pas vrai ?

Cependant, nos pages sont encore un peu mornes. Il serait bien d'y rajouter un peu de vie. Dans cette partie on va voir comment ajouter des éléments média comme des images, des vidéos et du son.

Les images

Les formats d'images

Comme vous le savez sûrement déjà, il existe plusieurs formats d'images mais quelles sont leurs spécificités et comment les utiliser ?

Tous les formats utilisent la **compression** pour réduire la taille des images sur le serveur. En effet, une image est constituée de millions de pixels chacun caractérisé par une couleur. Cela représente une très grosse quantité d'information à stocker que l'on veut réduire au maximum.

Dans le cadre d'un site internet, on va utiliser ces trois formats :

- Le **format JPEG** (*Joint Photographic Experts Group*) : à utiliser pour les **photographies**. En effet le format JPEG permet une compression optimale des photos. L'extension de l'image au format JPEG est **.jpeg** ou **.jpg** (cela dépend des cas).
- Le **format PNG** (*Portable Network Graphics*) : à utiliser pour **tout autre type d'image figée** (icônes, dessins, logo, graphiques, etc). L'avantage majeur du PNG est qu'il permet d'avoir un fond transparent, c'est très utile pour ne pas avoir de fond blanc indésirable qui masque l'arrière-plan. L'extension de l'image au format PNG est **.png**.
- Le **format GIF** (*Graphics Interchange Format*) : à utiliser pour des **images animées**. Le format GIF permet de regrouper plusieurs images en une seule. Ces images sont affichées les unes après les autres ce qui donne l'impression d'avoir une image animée. L'extension de l'image au format GIF est **.gif**.

L'insertion d'une image

Pour insérer une image dans votre code, nous utilisons la balise suivante :

```

```

Rappelez-vous, nous avions déjà vu cette balise précédemment dans le cours. Elle a deux attributs à mettre **A CHAQUE FOIS**.

L'attribut `src` indique le chemin relatif vers l'image à afficher alors que l'attribut `alt` donne une description **explicite** de l'image à afficher. L'attribut `alt` permet d'afficher le contenu de cet attribut si l'image ne s'affiche pas correctement, alimente les navigateurs pour malvoyant ou améliore le référencement de votre image sur les moteurs de recherche.

D'autres attributs existent mais leur usage est facultatif.

L'image cliquable

Il est facilement possible de créer une image cliquable en combinant les images et les liens :

```
<a href="vacances_oleron.jpg"></a>
```

Ici, on a un lien qui contient notre image en version miniature, c'est elle qui est affichée à l'écran. Lors du clic de la souris sur la petite image, le lien redirige vers notre image plus grande.

INSERER CAPTURE D'ECRAN

Le son

L'audio est une fonctionnalité implémentée avec HTML5. C'est donc une possibilité relativement récente qui peut ne pas être reconnue par des navigateurs web obsolètes ou non à jour. Avant, il fallait utiliser des plugins créés spécifiquement pour implémenter de l'audio.

Les formats de son

Tout comme les images, il existe plusieurs formats de son dont voici les principaux :

- **MP3** : il s'agit d'un des plus anciens et des plus connus. Comme il est ancien, il est compatible quasiment partout.
- **OGG** : format d'audio libre de droit donc assez utilisé sur Linux.
- **AAC** : format principalement utilisé par Apple.

L'insertion du son

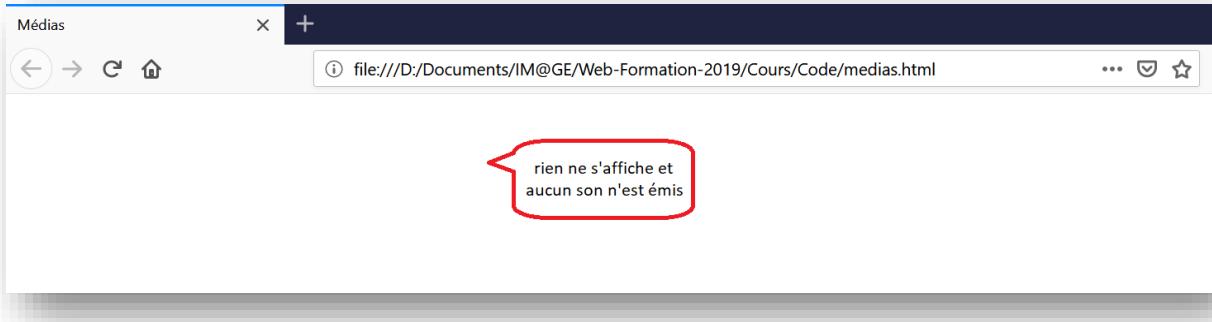
Pour insérer du son, on utilise la balise suivante :

```
<audio src="ma_musique.mp3"></audio>
```

Nous avons donc téléchargé une musique gratuite et **libre de droit** sur le site *musicscreen.be* : Il s'agit du titre *Grandioso* composé par Hicham Chahidi :

```
<audio src="Audio/Grandioso/Grandioso.mp3"></audio>
```

On obtient la page suivante et aucun son n'est émis :

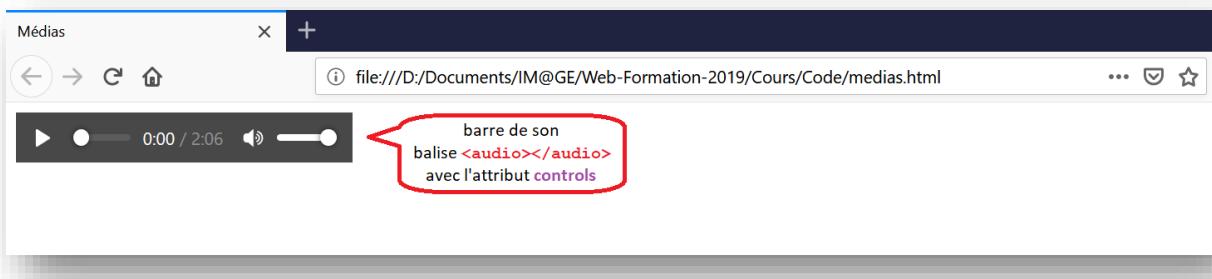


En effet, la balise audio indique seulement au navigateur web de charger les métadonnées du fichier audio.

En pratique, on utilise donc toujours la balise `<audio></audio>` avec l'attribut `controls` :

```
<audio src="Audio/Grandioso/Grandioso.mp3" controls></audio>
```

On a alors l'apparition d'une barre de son qui permet de gérer le lancement/l'arrêt de la musique, le défilement ainsi que le volume :



Pourquoi avoir fait une balise ouvrante et une balise fermante alors que l'on n'écrit rien entre les deux ? On aurait pu faire une balise orpheline !

La raison est qu'on nous laisse la possibilité d'écrire un message entre les deux balises dans le cas où la barre de son ne s'affiche pas correctement.

```
<audio src="Audio/Grandioso/Grandioso.mp3" controls>La barre de son ne s'affiche pas correctement, votre navigateur est-il à jour ?</audio>
```

Il y a également d'autres attributs :

- `width` : longueur de la barre de son
- `loop` : joue le son en boucle
- `autoplay` : lance le son automatiquement

Attention cependant à utiliser ces attributs avec parcimonie car le son peut vite devenir intrusif et peut être très agaçant pour l'utilisateur s'il lui est imposé. On évitera donc souvent d'utiliser `loop` et `autoplay`.

De plus, les navigateurs ont tous leurs spécificités et c'est également le cas avec les formats audio. Pour s'assurer du bon fonctionnement des musiques sur notre site, on va préférer mettre plusieurs formats de musique pour une même barre de son. Le navigateur essaiera de lancer le premier. S'il n'est pas compatible avec ce format, il essaiera avec le second, etc :

```

<audio controls>
    <source src="Audio/Grandioso/Grandioso.mp3" />
    <source src="Audio/Grandioso/Grandioso.ogg" />
    <source src="Audio/Grandioso/Grandioso.aac" />
</audio>

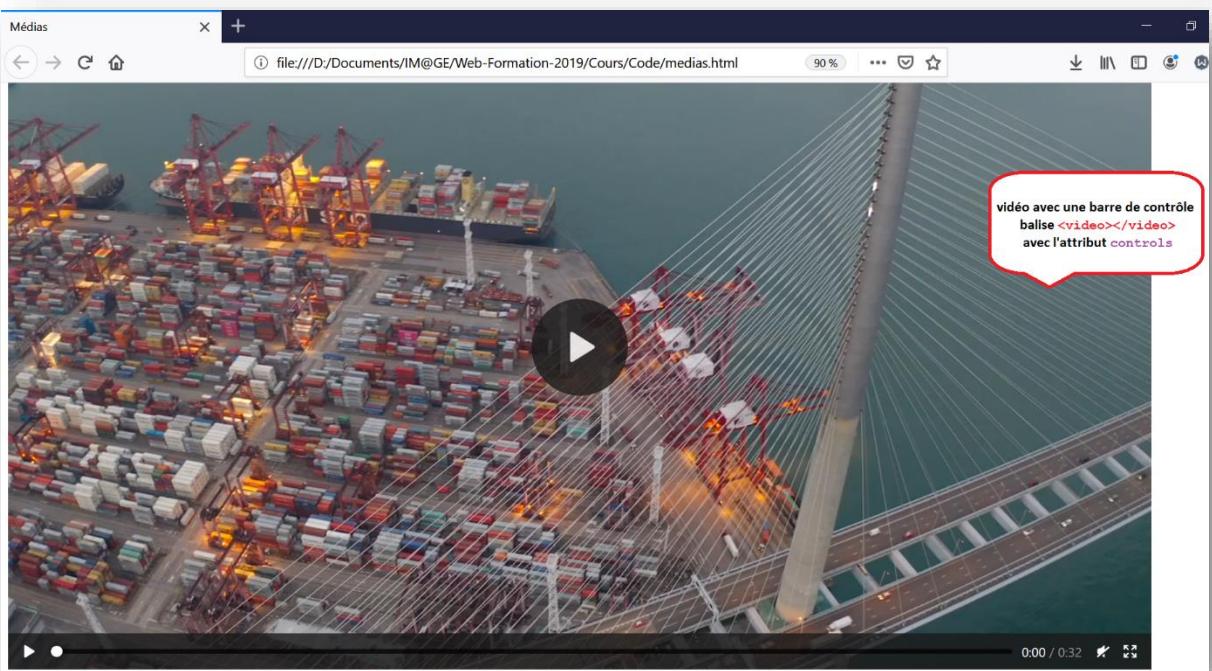
```

La vidéo

Désormais, on veut intégrer une vidéo dans notre page web, cela ressemble beaucoup à la syntaxe de l'audio :

```
<video src="Video/Bridge.mp4" controls />
```

On obtient :



Là aussi, il y a bien d'autres attributs dont en voici quelques-unes :

- **width** : longueur de la vidéo
- **height** : hauteur de la vidéo
- **loop** : joue le son en boucle
- **autoplay** : lance le son automatiquement
- **poster** : définit l'image affichée lorsque la vidéo n'est pas lancée. Par défaut le navigateur prend la toute première image qui compose la vidéo.

Encore une fois, utilisez **loop** et **autoplay** avec parcimonie !

Les figures

Une figure est un élément illustratif de votre page web. C'est un conteneur qui contient donc en général une image, une vidéo ou du texte. L'intérêt de la figure est de pouvoir ajouter une légende avec la balise **<figcaption></figcaption>**.

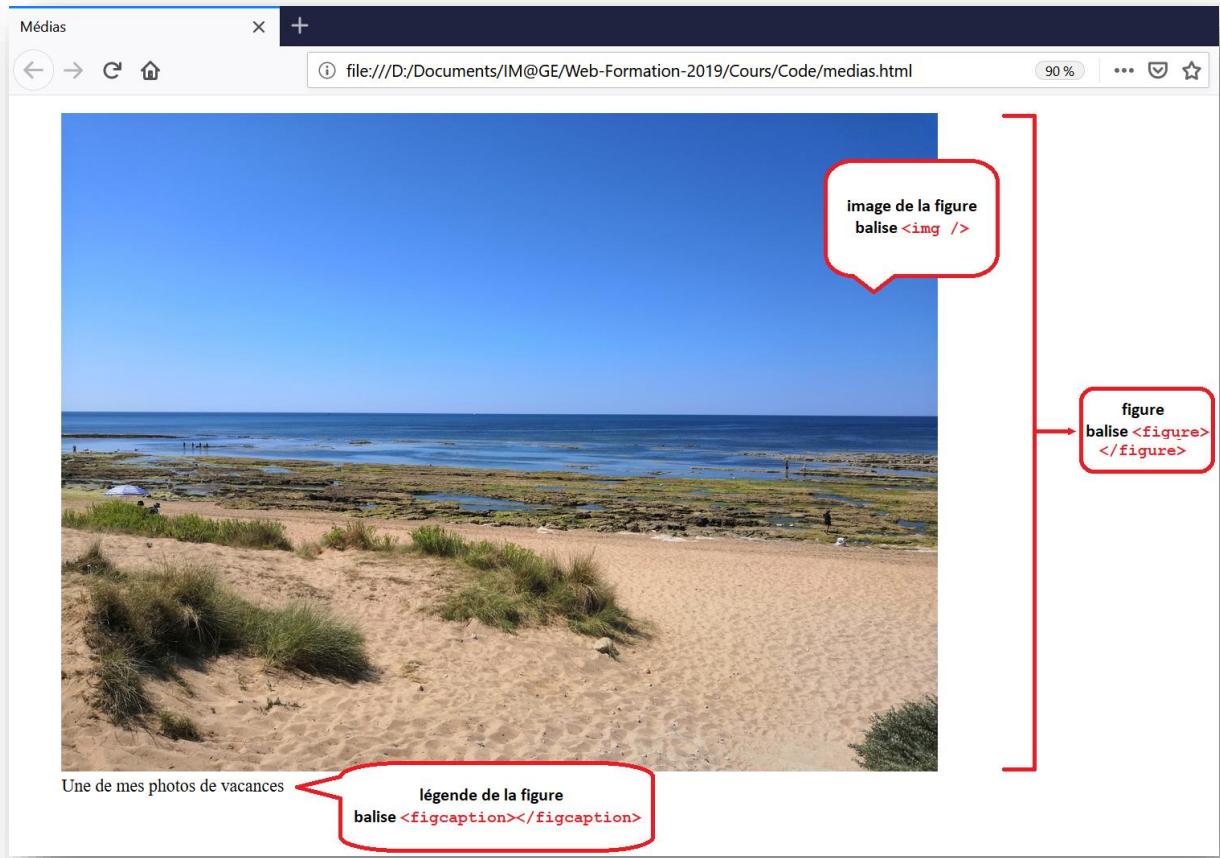
```
<figure>
```

```


<figcaption>Une de mes photos de vacances</figcaption>
</figure>

```

Ce qui donne :



Les formulaires

Les formulaires sont des éléments HTML qui permettent une interaction de votre site web avec l'utilisateur. Ce dernier peut entrer des informations dans des champs de recherche, répondre à un sondage avec des réponses uniques, faire un qcm à choix multiples, etc. Les possibilités sont infinies !

Mettre en place un formulaire

Pour insérer un formulaire on utilise la syntaxe suivante :

```

<form method="post" action="trait_form.php">
  <p>
    <!-- insérer ici les éléments de votre formulaire (champs de saisie, boutons, etc) --&gt;
  &lt;/p&gt;
&lt;/form&gt;
</pre>

```

Comme vous pouvez le constater, un formulaire se crée grâce à la balise `<form></form>`. Jusque-là rien de compliqué. Ce qu'il faut comprendre ensuite, c'est que comme un formulaire est fait pour interagir avec l'utilisateur, il va falloir **récupérer les informations saisies** et puis **réaliser un traitement** dessus. On va par exemple pouvoir enregistrer les réponses dans une base de données, afficher un message à l'écran, etc.

Pour cela, on doit spécifier quelle méthode utiliser pour récupérer les données et les envoyer à notre fichier qui les traitera, c'est le rôle de l'attribut `method`.

Il y a deux méthodes possibles :

- la méthode `get` : les données sont envoyées dans l'URL de la page. Cette méthode limite l'information envoyée à 255 caractères. On peut l'utiliser pour des données courtes et sans vraiment d'importance puisqu'elles sont visibles dans l'URL (ne pas utiliser `get` pour récupérer un mot de passe par exemple).
- la méthode `post` : c'est la plus couramment utilisée car elle n'est pas limitée en quantité d'information à envoyer au fichier de traitement.

Dans l'exemple ci-dessus j'ai décidé d'utiliser le cas le plus courant, c'est-à-dire la méthode `post`.

Une fois la méthode renseignée, on peut récupérer les informations mais encore faut-il avoir indiqué un fichier de traitement à qui les envoyer. C'est ce dont s'occupe l'attribut `action`. Le fichier vers lequel nous envoyons les informations est un fichier un peu spécial puisqu'il s'agit d'un fichier **PHP**.

En effet, rappelez-vous ! Le HTML et le CSS sont des langages utilisés pour le front-end, le design de nos pages et leur structure. Pour tout ce qui attrait aux traitements, c'est le back-end qui s'en charge. Le PHP est en fait un langage utilisé pour le back-end. Mais laissons cela de côté pour le moment. Nous reviendrons au sujet du traitement des formulaires dans la partie du cours sur le back-end.

Remarque : Tous les éléments d'un formulaire sont toujours **encapsulés dans un et un seul paragraphe** !

Désormais, concentrons-nous sur les différents éléments que nous pouvons utiliser dans nos formulaires.

Les champs de saisie

Les champs de saisie se caractérisent par la balise suivante : `<input />`.

Cette balise crée un élément qui va permettre de laisser à l'utilisateur la possibilité d'entrer des données. Le type de champs de saisie sera déterminé par l'attribut `type`.

Le champ de texte

Pour un champ de texte, on utilise l'attribut `type="text"` :

```
<form method="post" action="trait_form.php">
    <p>
        <label>Nom d'utilisateur :</label>
        <input type="text" />
    </p>
</form>
```

Un champ de saisie, et cela ne vaut pas que pour les champs de texte mais pour tout type de champs de saisie, est généralement accompagné d'un label descriptif. Pour lier le label au champ de saisie, on utilise l'attribut `id` sur le champ lui-même pour l'identifier et l'attribut `for` sur le label pour le référencer. Ainsi, désormais si vous cliquez sur le label un curseur apparaît dans le champ correspondant.

De manière générale, on va également donner un nom à notre champ qui sera rattaché à la donnée retournée par le formulaire. Cela permet ensuite au niveau du back-end de récupérer cette donnée plus facilement. Pour ce faire, on renseigne l'attribut `name`.

```
<form method="post" action="trait_form.php">  
    <p>  
        <label for="username">Nom d'utilisateur :</label>  
        <input type="text" id="username" name="username_result" />  
    </p>  
</form>
```

Résultat :



La balise `<input></input>` possède de nombreux autres attributs dont en voici quelques-uns (comme d'habitude n'hésitez à fouiner par vous-même pour en savoir plus sur les autres attributs) :

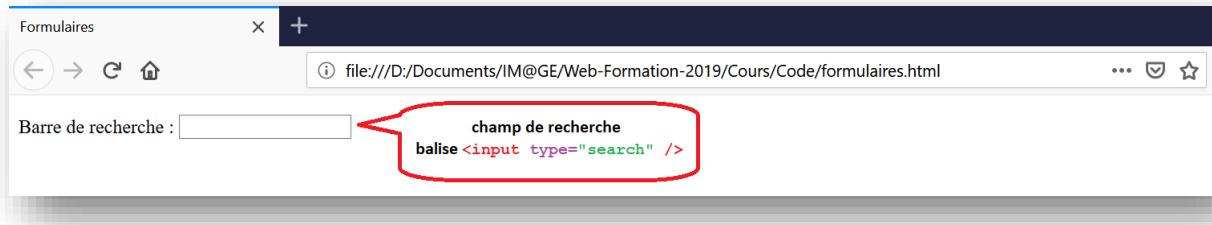
- `placeholder` : écrit une indication dans le champ qui disparaît lors du clic de la souris.
- `value` : met une valeur par défaut qui ne disparaît pas au clic de la souris. Il s'agit de la valeur associée à la variable ayant pour nom `name`.
- `size` : indique la longueur du champ affiché à l'écran en nombre de caractères à saisir.
- `maxlength` : donne le nombre maximal de caractères à saisir dans le champ.
- `disabled` : grise un champ.

Le champ de recherche

On utilise l'attribut `type="search"`.

```
<form method="post" action="trait_form.php">  
    <p>  
        <label for="searchbar">Barre de recherche :</label>  
        <input type="search" id="searchbar" name="search_result" />  
    </p>  
</form>
```

Résultat :



Sur mon écran, rien a changé par rapport au champ de texte classique ?!

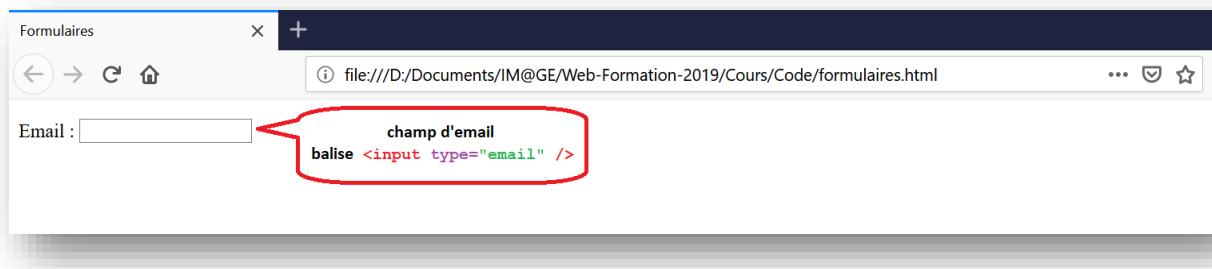
En effet, cela ne veut pas dire que cela ne sert à rien pour autant et qu'il faut mettre un champ de texte pour faire une barre de recherche ! Le navigateur va comprendre que l'input que vous avez créé est une barre de recherche. Certains navigateurs ajoutent une loupe dans le champ.

Le champ d'email

On utilise l'attribut `type="email"` :

```
<form method="post" action="trait_form.php">
  <p>
    <label for="email">Email :</label>
    <input type="email" id="email" name="email_result" />
  </p>
</form>
```

Cependant, à première vue rien n'a changé :



Le navigateur a cependant interprété cette balise comme un champ de saisie d'email. Il peut alors agir en conséquence. Mozilla Firefox, par exemple, fait une vérification sommaire de la validité de l'adresse email (il regarde seulement si le résultat saisi contient un caractère « @ ») :

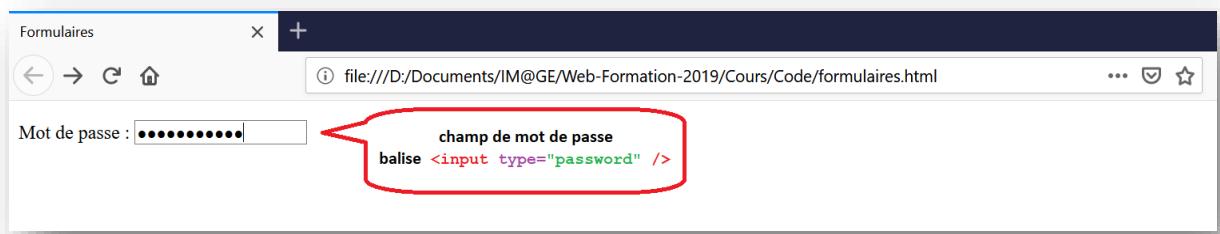


Le champ de mot de passe

On utilise l'attribut : `type="password"`.

```
<form method="post" action="trait_form.php">
    <p>
        <label for="password">Mot de passe :</label>
        <input type="password" id="password" name="password_result" />
    </p>
</form>
```

Les caractères saisis sont alors masqués à l'écran :



Le champ d'URL

On utilise l'attribut `type="url"`.

```
<form method="post" action="trait_form.php">
    <p>
        <label for="url">Entrer une URL :</label>
        <input type="url" id="url" name="url_result" />
    </p>
</form>
```

Le champ sur PC est inchangé, cet attribut permet en revanche de modifier automatiquement le clavier des smartphones avec des raccourcis pour saisir une URL. Vous avez maintenant compris le principe, le navigateur réalise des vérifications élémentaires :



Le champ de nombre

On utilise l'attribut `type="number"`.

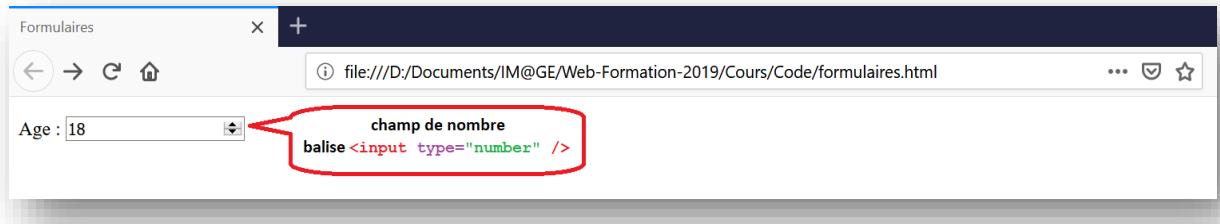
```
<form method="post" action="trait_form.php">
    <p>
```

```

<label for="age">Age :</label>
<input type="number" id="age" name="age_result" />
</p>
</form>

```

On a alors un champ de saisie un petit peu différent, puisqu'il possède des flèches vers le haut et vers le bas ou incrémenter ou décrémenter notre nombre entier :



Le champ de numéro de téléphone

On utilise l'attribut `type="tel"`.

```

<form method="post" action="trait_form.php">
    <p>
        <label for="tel">Téléphone :</label>
        <input type="tel" id="tel" name="tel_result" />
    </p>
</form>

```



Le curseur

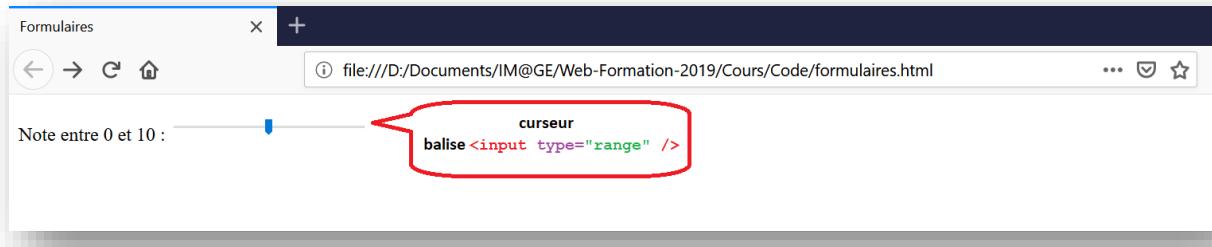
On utilise l'attribut `type="range"`.

```

<form method="post" action="trait_form.php">
    <p>
        <label for="mark">Note entre 1 et 10 :</label>
        <input type="range" id="mark" name="mark_result" min="0" max="10"
               step="1" />
    </p>
</form>

```

On obtient un curseur, dont le minimum est 0 et le maximum est 10 et que l'on peut régler avec un pas de 1:

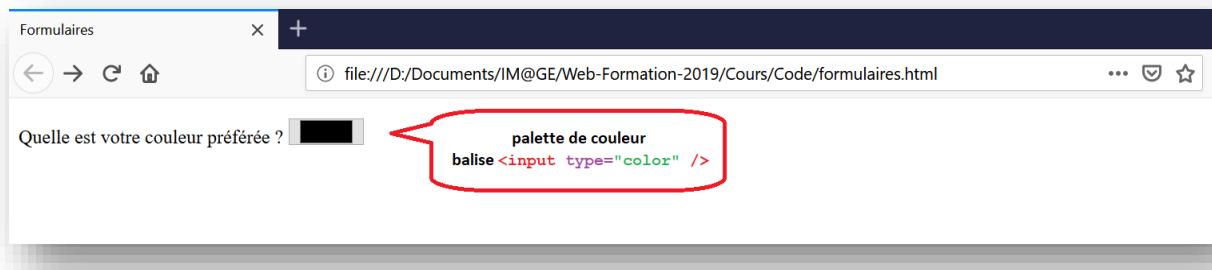


La palette de couleur

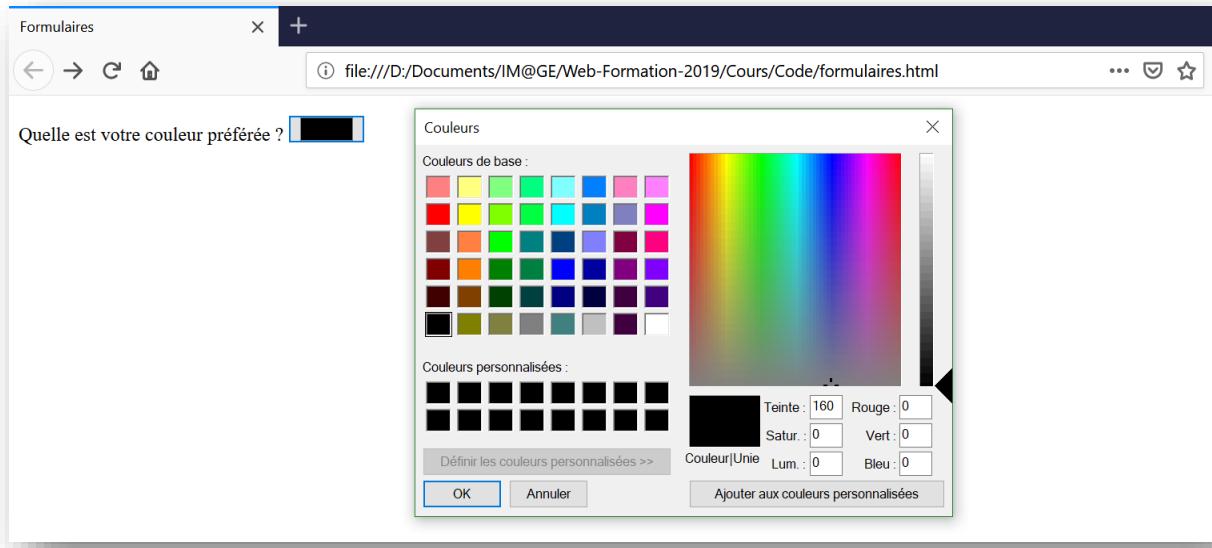
On utilise l'attribut `type="color"`.

```
<form method="post" action="trait_form.php">
    <p>
        <label for="color">Quelle est votre couleur préférée ?</label>
        <input type="color" id="color" name="color_result"
value="#000000"/>
    </p>
</form>
```

On a alors un bouton contenant un rectangle noir :



Si l'on clique sur ce bouton, une palette de couleur apparaît et on peut alors choisir une couleur parmi toute la palette. Cette dernière est alors sélectionnée avec son code couleur. Nous verrons comment est codé la couleur dans le cours sur le CSS :



Le formulaire à choix unique

On utilise l'attribut `type="radio"`. Cependant, cela change un peu de ce que l'on a pu voir jusqu'ici puisqu'on n'a pas un élément mais plusieurs et l'utilisateur ne peut en cocher qu'un à la fois. Comme toujours, chaque élément à un `id` différent car un est `id` est par définition unique. Ensuite, comme on ne peut cocher qu'une case à la fois, on va indiquer que tous les éléments se réfèrent à la même variable. Comme la variable en question ne peut contenir qu'une valeur à la fois, on ne pourra alors sélectionner qu'un élément en même temps. Vous l'avez compris, on va donc mettre le même attribut `name` à tous les éléments. Enfin, comme les éléments se réfèrent à la même variable, on va distinguer quelle case a été cochée grâce à l'attribut `value`.

```
<form method="post" action="trait_form.php">
    <p>
        Combien d'heure de sport faites-vous par semaine en moyenne ?<br />
        <input type="radio" id="moins2h" name="freq_sport" value="moins2h" /><label for="moins2h">Moins de 2h</label>
        <input type="radio" id="2h_3h" name="freq_sport" value="2h_3h" /><label for="2h_3h">Entre 2h et 3h</label>
        <input type="radio" id="3h_4h" name="freq_sport" value="3h_4h" /><label for="3h_4h">Entre 3h et 4h</label>
        <input type="radio" id="plus4h" name="freq_sport" value="plus4h" /><label for="plus4h">Plus de 4h</label>
    </p>
</form>
```

Les boutons apparaissent sous la forme de petits cercles, vous pouvez constater qu'on ne peut cocher qu'une seule case à la fois. De telles cases sont appelées des **radio boutons** :

Formulaires x +

Combien d'heure de sport faites-vous par semaine en moyenne ?

Moins de 2h Entre 2h et 3h Entre 3h et 4h Plus de 4h

radio bouton
balise `<input type="radio" />`

Le formulaire à choix multiples

On utilise l'attribut `type="checkbox"`. Ici, c'est le cas précédent simplifié. En effet, comme on laisse la possibilité à l'utilisateur de donner plusieurs réponses en même temps, on donne un attribut `name` différent pour chaque élément. Ainsi, pas de besoin de renseigner d'attribut `value`.

```
<form method="post" action="trait_form.php">
<p>
    Quel type de vidéo YouTube regardez-vous :<br />
    <input type="checkbox" name="lifestyle" id="lifestyle" /> <label
    for="lifestyle">Lifestyle</label><br />
    <input type="checkbox" name="sport" id="sport" /> <label
    for="sport">Sport</label><br />
    <input type="checkbox" name="humour" id="humour" /> <label
    for="humour">Divertissement et humour</label><br />
    <input type="checkbox" name="beaute" id="beaute" /> <label
    for="beaute">Tuto beauté</label><br />
    <input type="checkbox" name="autre" id="autre" /> <label
    for="autre">Autre type de vidéo</label>
</p>
</form>
```

Cette fois-ci, les champs de saisie sont sous la forme de cases à cocher. Vous pouvez constater que l'on peut cocher plusieurs cases en même temps.

Formulaires x +

Quel type de vidéo YouTube regardez-vous :

Lifestyle
 Sport
 Divertissement et humour
 Tuto beauté
 Autre type de vidéo

case à cocher
balise `<input type="checkbox" />`

La liste déroulante

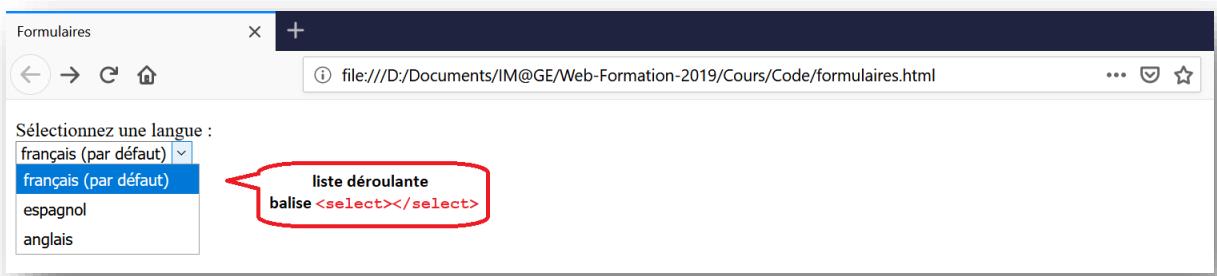
On utilise la balise `<select></select>`.

```

<form method="post" action="trait_form.php">
    <p>
        <label for="pays">Sélectionnez une langue :</label><br />
        <select name="pays" id="pays">
            <option value="francais">français (par défaut)</option>
            <option value="espagnol">espagnol</option>
            <option value="anglais">anglais</option>
        </select>
    </p>
</form>

```

On obtient alors une liste déroulante toute simple :



Cependant, nous aurions besoin d'une liste déroulante avec un niveau de détail beaucoup plus fin. Et bien cela est possible en regroupant nos différentes options dans des groupes :

```

<form method="post" action="trait_form.php">
    <p>
        <label for="langue">Sélectionnez une langue :</label><br />
        <select name="langue" id="langue">
            <optgroup label="européen">
                <option value="francais">français (par défaut)</option>
                <option value="espagnol">espagnol</option>
                <option value="anglais">anglais</option>
            </optgroup>

            <optgroup label="asiatique">
                <option value="mandarin">mandarin</option>
                <option value="japonais">japonais</option>
                <option value="russe">russe</option>
            </optgroup>
        </select>
    </p>
</form>

```

On obtient alors une liste déroulante avec des catégories :



La zone de texte multiligne

On utilise la balise `<textarea></textarea>`.

```
<form method="post" action="trait_form.php">
    <p>
        <label for="commentaire">Laissez un commentaire si vous le
        souhaitez :</label><br />
        <textarea name="commentaire" id="commentaire"></textarea>
    </p>
</form>
```



Le regroupement des différents champs

Si le cœur vous en dit, vous pouvez regroupez vos différents champs de saisies au sein des balises `<fieldset></fieldset>`. Cela permet de les regrouper dans un encadré. On peut également rajouter un titre à l'encadré avec la balise `<legend></legend>`:

```
<fieldset>
    <legend>Sondage</legend>

    <form method="post" action="trait_form.php">
        <p>
```

```

Combien d'heure de sport faites-vous par semaine en moyenne  

?<br />

<input type="radio" id="moins2h" name="freq_sport" value="moins2h" /><label for="moins2h">Moins de 2h</label>
<input type="radio" id="2h_3h" name="freq_sport" value="2h_3h" /><label for="2h_3h">Entre 2h et 3h</label>
<input type="radio" id="3h_4h" name="freq_sport" value="3h_4h" /><label for="3h_4h">Entre 3h et 4h</label>
<input type="radio" id="plus4h" name="freq_sport" value="plus4h" /><label for="plus4h">Plus de 4h</label>
</p>
</form>

<form method="post" action="trait_form.php">
<p>
    Quel type de vidéo YouTube regardez-vous :<br />
    <input type="checkbox" name="lifestyle" id="lifestyle" />
<label for="lifestyle">Lifestyle</label><br />
    <input type="checkbox" name="sport" id="sport" /> <label for="sport">Sport</label><br />
    <input type="checkbox" name="humour" id="humour" /> <label for="humour">Divertissement et humour</label><br />
    <input type="checkbox" name="beaute" id="beaute" /> <label for="beaute">Tuto beauté</label><br />
    <input type="checkbox" name="autre" id="autre" /> <label for="autre">Autre type de vidéo</label>
</p>
</form>
</fieldset>

```

On obtient alors le résultat suivant :

Formulaires

Sondage

Combien d'heure de sport faites-vous par semaine en moyenne ?

Moins de 2h Entre 2h et 3h Entre 3h et 4h Plus de 4h

Quel type de vidéo YouTube regardez-vous :

Lifestyle
 Sport
 Divertissement et humour
 Tuto beauté
 Autre type de vidéo

Les conteneurs

Bien ! Désormais, nous avons tous les éléments HTML susceptibles de nous intéresser. Nous allons maintenant pouvoir apprendre à structurer notre page HTML proprement.

Mais que veux-tu dire par structurer ?

Eh bien lorsque vous allez créer votre page, vous voudrez probablement dire dans votre code HTML : « ceci est un entête, ces éléments composent un pied de page, cela est une bannière, ceci est une section contenant l'ensemble de mes articles... ».

Pour cela, on va utiliser des balises structurantes.

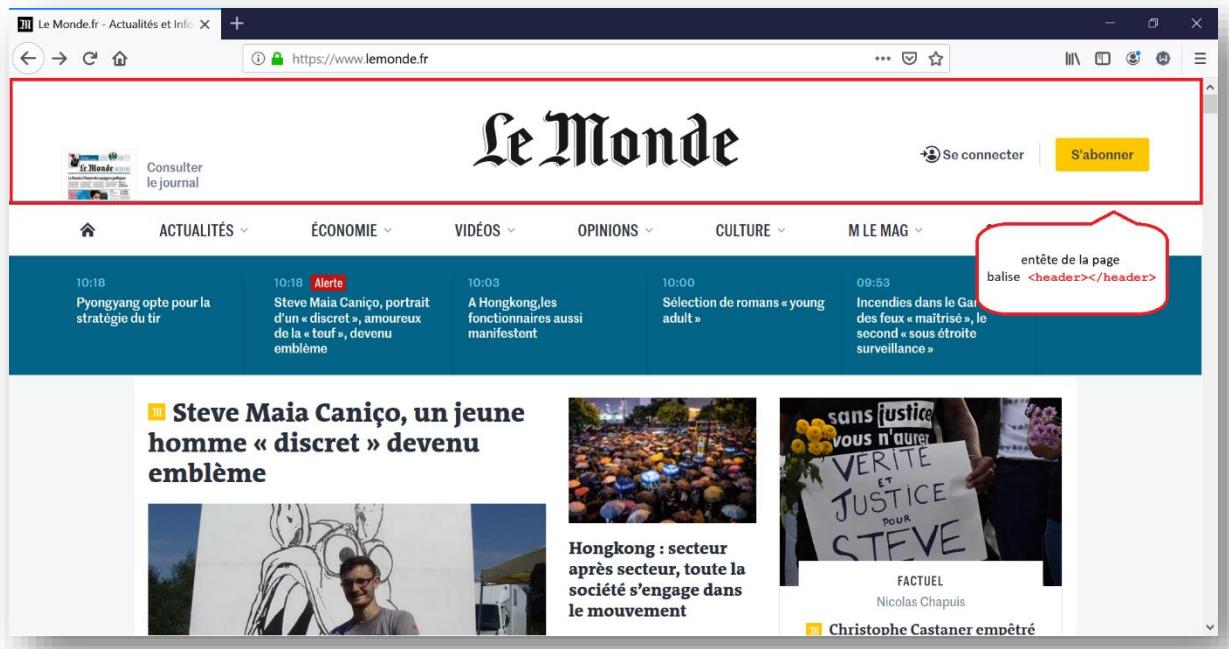
Les différents types de conteneurs

L'entête

On utilise la balise `<header></header>`. L'entête contient généralement le titre du site, un logo, une bannière. Il peut également contenir un menu.

```
<header>
    <!-- Ici se trouve le contenu d'un entête -->
</header>
```

Dans l'exemple ci-dessous, on considère que la navigation du site ne fait pas partie de l'entête :



Le pied de page

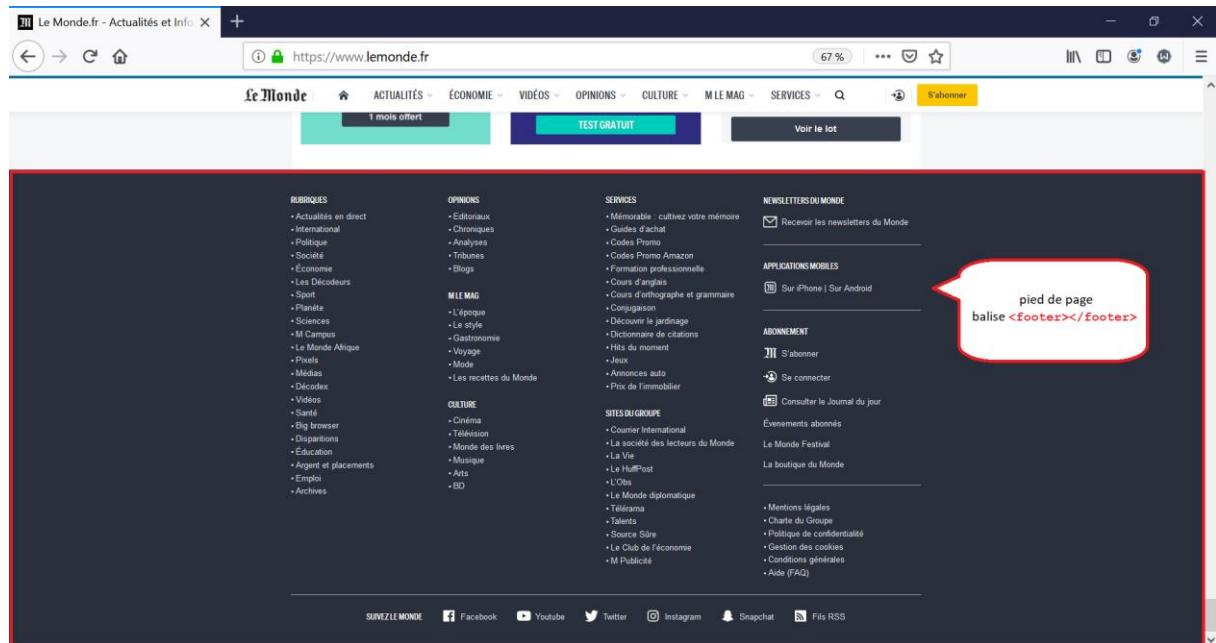
On utilise la balise `<footer></footer>`. Elle contient généralement des informations sur le site : des coordonnées pour joindre les administrateurs du site, des informations sur l'entreprise qui gère le site

(adresse, statut de l'entreprise, etc), les CGU (Conditions Générales d'Utilisation), les liens vers les réseaux sociaux, etc.

<footer>

<!-- Ici se trouve le contenu d'un pied de page -->

</footer>



La section

On utilise la balise **<section></section>**. Cette dernière est généralement utilisée pour délimiter les zones qui contiennent des informations sur différents thèmes.

<section>

<!-- Ici se trouve le contenu d'une section -->

</section>

Dans l'exemple ci-dessous, chaque section délimite un résumé d'un article, chacun traitant d'un thème différent :

The screenshot shows a news article titled "La menace de Pékin aux manifestants à Hongkong : « Ceux qui jouent avec le feu péirront par le feu »". The article includes a photograph of protesters in Hong Kong. To the right, there are several sidebar elements: a thumbnail for "Pourquoi Keanu Reeves obsède Internet", another for "« La Hulotte » : la vie prisée des animaux", and a section titled "« La voiture est devenue un élément essentiel de la dignité du citoyen »". A red callout box highlights the word "article" in the sidebar text.

L'article

On utilise la balise `<article></article>`. Son nom est assez explicite, elle contient le contenu d'un article.

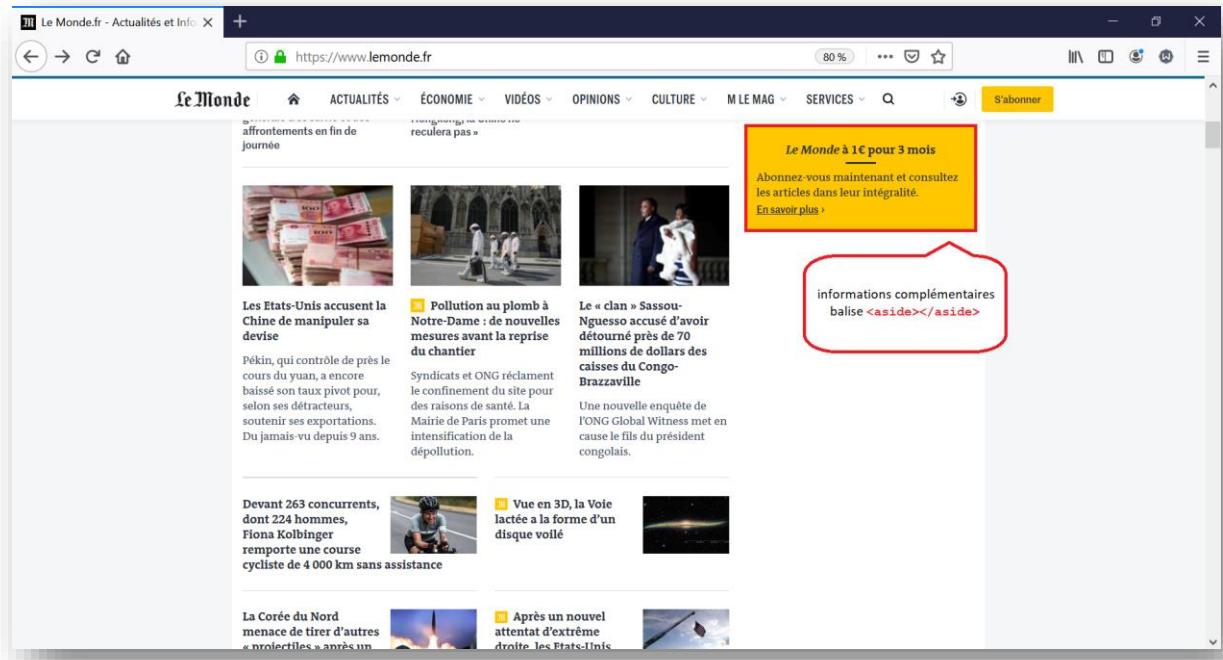
```
<article>
  <!-- Ici se trouve le contenu d'un article -->
</article>
```

The screenshot shows an article from Le Monde titled "Pourquoi Keanu Reeves obsède Internet". The article is by Pauline Croquet and includes a "DÉCRYPTAGES" section. A red callout box highlights the word "article" in the sidebar text.

Les informations annexes

On utilise la balise `<aside></aside>`. Cette dernière contient des informations complémentaires. Elle est généralement située sur un côté de la page puisque le contenu général est bien souvent à gauche de la page.

```
<aside>
  <!-- Ici se trouvent les informations annexes -->
</aside>
```



La navigation

On utilise la balise `<nav></nav>`. On y place des menus, sommaires, liens en tout genre.

```
<nav>
  <!-- Ici se trouvent les éléments de navigation -->
</nav>
```

The screenshot shows the Le Monde website homepage. At the top, there is a navigation bar with links for 'ACTUALITÉS', 'ÉCONOMIE', 'VIDÉOS', 'OPINIONS', 'CULTURE', 'M LE MAG', 'SERVICES', and a search icon. A red box highlights this navigation bar. Below the navigation bar, there are several news cards. One card for 'Steve Maia Caniço' is highlighted with a red box and a callout bubble containing the text 'barre de navigation balise <nav></nav>'.

La division du contenu

On utilise la balise `<div></div>`. Cette balise n'a pas un sens aussi marqué que les autres conteneurs, il permet simplement de séparer du contenu en plusieurs blocs distincts. L'intérêt principal est qu'on peut lui rajouter une classe CSS. Vous comprendrez mieux où je veux en venir par la suite, quand nous aborderons le cours sur le CSS.

```
<div>
    <!-- Ici se trouve le contenu d'un conteneur quelconque --&gt;
&lt;/div&gt;</pre>
```

The screenshot shows the Le Monde.fr website homepage. At the top, there's a navigation bar with links for 'Le Monde.fr - Actualités et Info', 'Consulter le journal', 'Se connecter', and 'S'abonner'. Below the header, there's a main menu with categories like 'ACTUALITÉS', 'ÉCONOMIE', 'VIDÉOS', 'OPINIONS', 'CULTURE', 'M LE MAG', and 'SERVICES'. The main content area features a news feed with several articles. One article is highlighted with a red callout box containing the text: 'division du contenu balise <div></div>'. The visible news items include:

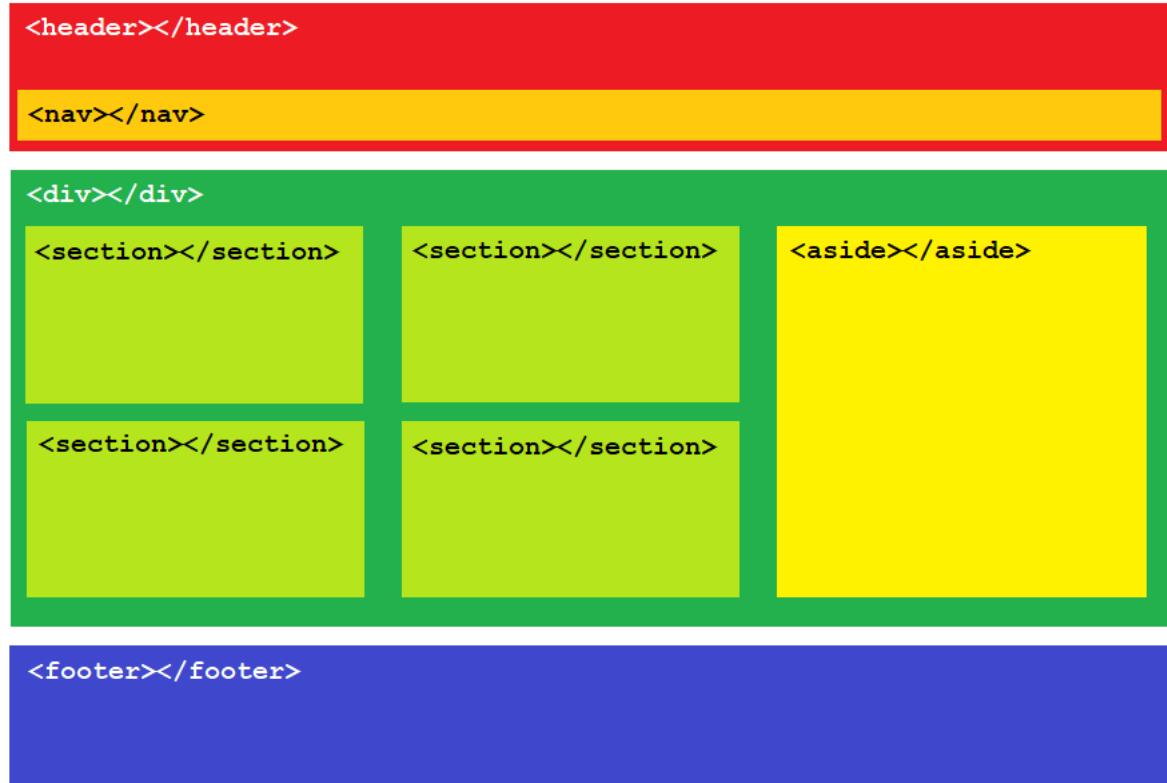
- 11:06 Football : le mercato son plein
- 11:05 Sibérie : explosion d'un dépôt de munitions
- 10:22 Alerté A chaque numéro, un animal ou une plante traitée comme un héros : « La Hulotte », le « journal le plus lu dans les terrains » depuis quarante-sept ans
- 09:55 Hongkong : Pékin menace fermement les manifestants
- 09:39 Mamoudou Barry inhumé dans son village de Guinée

Below the news feed, there are two main columns of articles. The left column features an article titled 'La menace de Pékin aux manifestants à Hongkong : « Ceux qui jouent avec le feu périront par le feu »' with a sub-image of a city skyline. The right column features an article titled 'La guerre des monnaies, riposte de la Chine à Trump' with a sub-image of Chinese flags and a portrait of a man. Further down, there are more articles, including one by Nathaniel Herzberg with the title 'A chaque numéro, un animal ou une plante traité comme un'.

Le modèle des boîtes

Nous avons pu voir les différents types de conteneurs existants. Cependant, je vous les ai présentés indépendamment les uns des autres. Hors, l'avantage des conteneurs est qu'ils sont emboitables les uns dans les autres, c'est ce que l'on appelle **le modèle des boîtes**.

Ce système d'emboitement laisse donc une infinité de possibilités au développeur pour organiser sa page comme il le souhaite. Voici un exemple de structuration d'une page web sous forme schématique :



Vous comprenez mieux désormais, pourquoi on appelle cela le modèle des boîtes ?

Rappel des bonnes pratiques en HTML

Faisons un rapide rappel des bonnes habitudes à prendre et des travers à éviter en HTML :

- Utiliser le HTML comme un langage qui définit la structure du HTML et rien d'autre (surtout pas sa mise en page) !

Exemple : Utiliser **** pour montrer qu'un mot est très important, ne pas l'utiliser pour mettre du texte en gras !

- Ecrire des commentaires si certains passages du code peuvent prêter à confusion.
- Utiliser les bonnes balises au bon endroit !

Exemple : Si on écrit un article, on ne mettra pas le contenu dans cet article dans une balise **<div></div>** mais dans une balise **<article></article>**. Ça paraît bête comme ça, mais vous verrez que l'on n'y pense pas toujours.

- Utiliser toujours des musiques, photos, vidéos **libres de droits** ! Il faut se renseigner car parfois il est également nécessaire de citer les ayants droits. Faites très attention à cela surtout si votre site est à but commercial ou génère des revenus. C'est de plus en plus contrôlé et vous pouvez avoir des problèmes, le vol de contenu n'est pas anodin.

- Eviter de lancer des musiques et de vidéos automatiquement. Je ne sais pas si ça vous est déjà arrivé, mais c'est très intrusif et cela peut devenir rapidement pénible ! L'objectif n'est pas de faire fuir les utilisateurs !
- Pour écrire du texte, utiliser toujours la balise de paragraphe `<p></p>`, la balise `<label></label>` est réservée pour légender des éléments de l'interface utilisateur comme les champs de saisie typiquement.

CSS

Le CSS, rappelez-vous, est le langage qui permet de réaliser la mise en forme de nos pages.

La syntaxe de base du CSS

Le CSS, tout comme le HTML est un langage plutôt simple à appréhender puisqu'il est très intuitif. Le CSS repose principalement sur trois éléments que l'on va utiliser constamment :

- **Le sélecteur** : c'est ce qui permet de sélectionner ce que l'on va modifier. Par exemple, si l'on décide de modifier l'aspect de nos paragraphes, le sélecteur fera référence à la balise correspondant au paragraphe en HTML.
- **La propriété** : c'est ce que l'on souhaite modifier sur notre élément. Par exemple, il peut s'agir de sa couleur, de sa taille, de sa longueur, etc.
- **La valeur** : il s'agit de la valeur que l'on assigne à une propriété. Par exemple, la valeur de la couleur peut être bleu, rouge ou encore vert.

Trois manières d'écrire du code CSS

Il existe trois façons différentes d'écrire du code CSS :

- Écrire dans une balise HTML avec l'attribut `style` : très mauvaise façon de faire.
- Écrire dans l'entête du fichier HTML (entre les balises `<head></head>`) : mieux mais pas encore la meilleure.
- Écrire dans un fichier CSS séparé du code HTML : beaucoup mieux !

L'écriture directement dans une balise HTML

Il est possible de rajouter un attribut style dans une balise HTML. Cela permet d'écrire le code CSS au sein de cet attribut. Cela se présente comme suit :

```
<balise style="propriété : valeur;"></balise>
```

Par exemple, si l'on veut mettre un titre en rouge :

```
<h1 style="color : red;">Mon grand titre</h1>
```

Testez ce code, vous verrez cela fonctionne !

Mais alors, pourquoi me le déconseilles-tu ?

Imaginons que l'on ait quinze titres sur notre page, il faudra rajouter la couleur quinze fois dans le code HTML ! Mettons que vingt minutes après, la couleur ne vous plaise plus et que vous voulez les mettre en bleu, je peux vous assurer que vous allez soupirer en modifiant tous les styles un à un ! En plus, cela surcharge votre code HTML inutilement.

Cette manière de faire est extrêmement archaïque et pénible, je vous la déconseille vivement. Cependant, c'est toujours bien de savoir qu'elle existe.

La seule utilité serait dans le cas où on a une modification spécifique à faire sur un ou deux éléments de la page HTML. A titre personnel je préfère cependant utiliser un identifiant sur cet élément (vous savez, c'est ce qu'on a utilisé pour identifier un élément lors de la création d'une ancre) et modifier cet élément avec la troisième méthode que nous verrons dans quelques minutes.

En résumé, n'utilisez jamais cette méthode !

L'écriture dans l'entête du fichier HTML

La seconde méthode consiste à écrire le code CSS dans l'entête du fichier HTML, c'est-à-dire entre les balises `<head></head>`.

Pour indiquer que le code en question est du CSS, on va rajouter les balises `<style></style>`. Ensuite, on va pouvoir écrire notre code CSS en utilisant des **sélecteurs**. Un sélecteur permet de désigner un ou plusieurs éléments dont on veut modifier des **propriétés** en leur assignant une certaine valeur. On va l'utiliser de cette façon :

```
selecteur {  
    propriété1 : valeur1;  
    propriété2 : valeur2;  
    <!-- Et ainsi de suite --&gt;<br/>}
```

Si on reprend notre exemple ci-dessus, pour mettre tous nos gros titres en rouge on procédera ainsi :

```
<!DOCTYPE html>  
  
<html>  
    <head>  
        <meta charset="utf-8" />  
        <title>Mes grands titres personnalisés</title>  
        <style>  
            h1 {  
                color: red;  
            }  
        </style>  
    </head>  
  
    <body>  
        <h1>Mon premier grand titre</h1>  
        <h2>Mon second grand titre</h2>  
        <h3>Mon troisième grand titre</h3>  
    </body>  
</html>
```

C'est déjà mieux vous en conviendrez ! Avec une seule instruction, on arrive à modifier tous les éléments du même type d'un coup. Cette méthode est utilisée lorsque l'on veut appliquer un certain style à une page en particulier (une page d'erreur par exemple). Cependant, ce n'est pas la technique la plus répandue car elle reste imparfaite.

En effet, dans le cadre de la mise en forme d'une page web, cette technique fonctionne parfaitement. Mais un site internet comporte en réalité des dizaines de pages et on ne veut pas recommencer la mise en forme à chaque nouvelle page créée. Ce que l'on souhaite en revanche, c'est d'appliquer un style à plusieurs pages d'un coup. Pour ce faire, direction la troisième méthode !

La création d'une feuille de style

Cette méthode est la plus répandue et celle que je vous recommande. En effet, il s'agit de la manière de faire la plus propre. Elle consiste à créer une feuille de style de manière à regrouper la mise en page au même endroit.

Mais qu'est-ce qu'une feuille de style au juste ?

Ce n'est ni plus ni moins qu'un fichier CSS. On appelle ce fichier ainsi car modifier l'aspect d'un élément revient à lui appliquer un style.

Pour pouvoir utiliser cette méthode il faut mettre en place deux étapes :

- Créer la feuille de style
- Relier la feuille de style au fichier HTML

Pour créer la feuille de style en elle-même, rien d'extraordinaire. Il suffit simplement de créer un fichier avec l'extension « .css » et de lui ajouter le code CSS que l'on souhaite. On va nommer notre feuille de style *mes_styles.css*. Ici, on veut mettre tous les gros titres en rouge donc le code ne change pas :

```
<!-- Contenu du fichier CSS -->
```

```
h1 {  
    color: red;  
}
```

Ensuite, pour relier le fichier CSS au code HTML, on retourne dans notre fameux code HTML et on lui rajoute une balise *<link />*. Cela semble assez naturel. On va par la suite lui rajouter des attributs : *rel* et *href*. Le premier indique que l'on met en relation le code HTML avec une feuille de style et le second donne le chemin vers le fichier à référencer :

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

Au final, le fichier HTML ressemble à ceci :

```
<!DOCTYPE html>  
  
<html>  
    <head>  
        <meta charset="utf-8" />  
        <link rel="stylesheet" href="mes_styles.css" />  
        <title>Mes grands titres personnalisés</title>  
    </head>  
  
<body>
```

```
<h1>Mon premier grand titre</h1>
<h2>Mon second grand titre</h1>
<h3>Mon troisième grand titre</h1>
</body>
</html>
```

L'avantage ici est que si l'on veut changer tous les titres de notre site, on n'a qu'à le modifier une fois dans le fichier CSS et tout est harmonisé partout. Le gain de temps est considérable !

Maintenant, imaginons que nous avons deux types de styles en fonction de nos pages web. Par exemple, concernant notre site d'articles que nous avions créé précédemment dans le cours. Nous aimerais avoir deux catégories d'articles, une catégorie « personnes inspirantes » dont fera partie nos articles sur Elon Musk, Steve Jobs, etc, et une catégorie « nouvelles technologies » qui regroupera des articles sur les nouveautés high-tech.

On décide alors que chaque catégorie d'article aura une mise en forme différente pour que cela soit plus personnalisé avec le thème en question.

Comment procéder pour mettre en place cette nouvelle mise en forme ?

Avec notre dernière technique, rien de plus simple ! Créons une nouvelle feuille de styles et on la reliera à tous nos articles traitant du sujet des nouvelles technologies. Avouez que c'est plus agréable que de modifier tous les éléments un par un !

Pour afficher votre page avec votre magnifique mise en forme, inutile d'ouvrir le fichier CSS, le fichier HTML suffit puisqu'on a préalablement relié le fichier CSS au fichier HTML.

Sélecteur et style

Comme vu précédemment, le sélecteur permet de sélectionner un certain type de balise pour lui appliquer un style. Ce sélecteur peut être différentes choses. Je vais vous expliquer en détail toutes les possibilités, vous allez voir que le langage CSS est extrêmement puissant puisqu'il permet de faire beaucoup de choses.

Les balises HTML

Commençons par le plus simple, les sélecteurs peuvent faire référence à une balise HTML. Ainsi, on peut modifier les propriétés de nos balises que nous avons soigneusement choisies. Comme vous pouvez maintenant le constater, il est important de bien structurer sa page HTML, quitte à y passer un peu de temps, pour pouvoir aller d'autant plus vite sur le CSS. Plus le HTML est bien réalisé, plus le CSS sera facile à mettre en place.

En HTML :

```
<p>Ceci est un paragraphe</p>
<p>Ceci est un autre paragraphe</p>
```

En CSS :

```
p {
    color: blue;
}
```

Les identifiants

L'inconvénient de mettre un style sur une balise HTML est que ce style sera appliqué sur toutes les balises ! Dans l'exemple ci-dessus, tous les paragraphes seront bleus. Hors on peut vouloir appliquer un style différent à un paragraphe qui résume un article et à un paragraphe qui présente son auteur par exemple.

Comment est-ce possible ?

Pour cela, le langage CSS introduit deux attributs qui peuvent être appliqués à n'importe quelle balise HTML : les identifiants et les classes.

Nous avions déjà vu les identifiants lors du passage sur les ancrès. L'identifiant permet de désigner de manière **unique** un élément HTML. A ce titre, deux balises HTML ne peuvent avoir le même identifiant. Au niveau du CSS, on peut modifier l'aspect d'un élément qui se réfère à un certain identifiant.

En HTML :

```
<p id="mon_paragraphe">Ceci est un paragraphe</p>
<p>Ceci est un autre paragraphe</p>
<p>Ceci est le dernier paragraphe</p>
```

En CSS :

```
p {
    color: blue;
}

#mon_paragraphe {
    color: red;
}
```

On voit que seul le paragraphe ayant pour id « mon_paragraphe » adopte une couleur rouge. Le second ainsi que le troisième restent bleus puisqu'il s'agit de paragraphes classiques. Renseigner un code CSS pour une balise HTML dans ce cas revient à faire une mise en forme par défaut.

Les classes

Cependant, les identifiants ne peuvent servir à modifier qu'un seul élément puisqu'ils sont par définition uniques.

Comment faire si l'on veut identifier plusieurs éléments pour modifier leur style ?

A ce moment-là entre en jeu la notion de classe. Une classe, c'est comme un identifiant sauf qu'elle peut être attribuée à plusieurs éléments.

En HTML :

```
<p class="author_desc">Ceci est un paragraphe</p>
<p class="author_desc">Ceci est un autre paragraphe</p>
```

```
<p>Ceci est le dernier paragraphe</p>
```

En CSS :

```
p {  
    color: blue;  
}  
  
.author_desc {  
    color: red;  
}
```

Vous noterez qu'en CSS, pour faire référence à une classe, on remplace le « # » par un « . ».

Les balises universelles

La balise ``

On sait comment modifier l'aspect de toutes les balises HTML d'un même type désormais et même en modifier seulement certaines. Cependant cela n'est pas suffisant. Imaginons le code suivant :

```
<p>Le bleu est ma couleur préférée</p>
```

J'aimerais mettre le mot « bleu » de la couleur bleue pour faire un effet esthétique sympas (je sais je suis un grand artiste ne soyez pas jaloux). Rien de ce que nous avons appris jusque-là ne nous permet pas de le faire sans bricoler un code bancal.

C'est là qu'intervient la balise universelle ``. Cette balise n'a aucune signification particulière, elle a juste été créée pour qu'on puisse lui assigner une classe :

HTML :

```
<p>Le <span class="blue_text">bleu</span> est ma couleur préférée</p>
```

CSS :

```
.blue_text {  
    color: blue;  
}
```

On dit que `` est une balise de type **inline** car elle est utilisée sur des lignes de texte. C'est un peu bancal comme définition mais nous y reviendrons bientôt, pas d'inquiétude !

La balise `<div></div>`

La balise `` est ce qui est utilisé pour du texte. Une autre balise reprend exactement le même principe mais pour les conteneurs. Cette balise nous l'avons déjà vue dans le chapitre sur les conteneurs, même si nous ne savions pas exactement à quoi cela correspondait. Il s'agit de la balise

<div></div>. Si vous vous remémorez ce passage du cours, tous les conteneurs avaient un sens précis alors que celui-là servait à « diviser du contenu ». C'est à peu de choses près le même rôle qu'on tous les conteneurs en fait. Le vrai rôle de la balise **<div></div>** est en fait d'isoler du contenu dans un bloc afin de structurer son code et de pouvoir attribuer des classes à ces différents conteneurs. C'est une balise très utilisée en CSS car elle permet d'accumuler un plus grand nombre de classes et donc d'enrichir la mise en page sur plusieurs niveaux.

On dit que la balise **<div></div>** est de type bloc car elle est utilisée pour créer des blocs d'éléments.

AJOUTER DU CODE HTML ET CSS

Les sélecteurs avancés

Tous les types de sélecteurs que nous avons vu précédemment sont plutôt sommaires. Il en existe une multitude, plus ou moins complexes, qui permettent d'obtenir un niveau de détail avancé dans la sélection d'un élément. Il en existe des dizaines, nous évoquerons seulement les principaux. Pour plus de renseignements, comme d'habitude, Google est votre ami ;)

Le sélecteur universel

Comme son nom l'indique, le sélecteur universel est utilisé pour sélectionner tous les éléments de la page :

CSS :

```
* {  
    background-color: grey;  
}
```

A ce moment-là, tous les éléments auront par défaut un fond gris.

Le sélecteur de plusieurs balises

Imaginons que nous ayons le code suivant :

CSS :

```
h1 {  
    color: green;  
}  
  
h2 {  
    color: green;  
}
```

Le code ci-dessus est alors équivalent à ceci :

```
h1, h2 {  
    color: green;  
}
```

Le sélecteur d'une balise qui en suit une autre

CSS :

```
h1 p {  
    color: purple;  
}
```

Dans l'exemple ci-dessus, tout paragraphe qui suit un gros titre sera mis en violet.

Le sélecteur d'une balise contenue dans une autre

CSS :

```
.citation p {  
    font-style: italic;  
}
```

Ainsi, tout paragraphe dans un élément ayant pour classe « citation » sera mis en italique.

Remarque : Les trois derniers type de sélecteurs ont été présenté sur un niveau de profondeur de deux étages. Cependant, il n'y a pas de limite et si l'on veut par exemple faire un sélecteur de plusieurs balises à n balises, rien ne nous en empêche.

Les commentaires

Les commentaires en CSS s'écrivent de la manière suivante :

```
/* Ceci est un commentaire */
```

Je vais vous épargner cette fois un long discours sur l'importance des commentaires, je vous l'ai assez rabâché dans le passage du cours concernant les commentaires en HTML. Passons donc à la suite sans plus attendre !

La couleur

La notion de couleur en informatique est plus complexe qu'il n'y paraît. Il y a différentes notations pour désigner une couleur, ces dernières ayant pour but de laisser la possibilité au développeur de disposer de la palette de couleur la plus large possible.

Les notations

Désignation d'une couleur par son nom

Le CSS a nommé certaines couleurs, les couleurs principales en somme, pour qu'elles soient facilement accessibles. Il y en a des dizaines, je vous invite à vous renseigner sur toutes les couleurs existantes.

En voici les principales :

- White
- Aqua
- Blue
- Navy

- Teal
- Green
- Lime
- Olive
- Maroon
- Beige
- Yellow
- Orange
- Red
- Purple
- Fushia
- Silver
- Gray
- Black

Cependant, cette façon de faire limite grandement les possibilités. Dans le monde réel, il y a des millions de couleurs, si ce n'est une infinité. Nommer une à une chaque couleur, chaque nuance, c'est mission impossible. On n'a donc mis en place des systèmes de codes couleur.

[La notation hexadécimale](#)

Nous avions déjà aperçu cette notation lors de la mise en place d'une palette de couleur en HTML, sans vraiment comprendre en quoi cela consistait. Une couleur en hexadécimal, cela a une tête comme ceci : #58FB26.

Ainsi le code CSS est le suivant :

```
p {
    color: #58FB26;
}
```

Il s'agit d'une notation hexadécimale, c'est-à-dire une notation en base 16. Les valeurs vont de 0 à 9 puis de A à F. En effet, les lettres de A à F remplacent les nombres de 11 à 16 pour avoir un code sur un seul caractère quelle que soit la valeur.

Il faut savoir que toutes les couleurs peuvent se retrouver par combinaisons du rouge, du vert et du bleu. Ainsi, la notation hexadécimale pour les couleurs fonctionne par paire de valeurs. Les deux premières valeurs codent le rouge, les deux suivantes le vert et les deux dernières le bleu.

On obtient alors 16 777 216 codes couleurs possibles, c'est déjà beaucoup mieux !

#000000 correspond à l'absence de couleur donc au noir tandis que #FFFFFF correspond à la présence de toutes les couleurs donc au blanc.

Pour obtenir les codes couleurs que vous souhaitez, ne vous cassez pas la tête dans des calculs, les logiciels font ça mieux que vous. Faites une rapide recherche sur Google et vous trouverez tout un tas de site que celui-ci : <https://htmlcolorcodes.com/fr/>.

[La notation RGB](#)

La notation RGB est strictement la même chose que la notation hexadécimale sauf que l'on note les valeurs en base 10 au lieu de les noter en base 16. En d'autres termes, au lieu d'avoir des valeurs allant de #00 à #FF on va avoir des valeurs comprises entre 0 et 255.

On utilise la syntaxe ci-dessous :

```
P {  
    color: rgb(88, 251, 36);  
}
```

Encore une fois, ne vous prenez pas la tête, aller sur un site tel que celui-ci pour réaliser vos codes couleurs : <https://htmlcolorcodes.com/fr/>.

Ajouter un fond

La couleur de fond

Nous avons vu dans de nombreux exemples que la couleur se fait avec la propriété `color`. Mais on peut également rajouter un fond à nos éléments avec la propriété `background-color`.

Par exemple :

```
body {  
    background-color: DarkOrange;  
}
```

L'image de fond

Ajouter de la couleur à notre fond c'est cool, mais on peut également y appliquer une image avec la propriété `background-img` :

```
body {  
    background-img: url("img/background.png");  
}
```

On peut utiliser une URL absolue avec « `http://mon_url` » ou relative avec le chemin relatif depuis votre fichier CSS (et non pas depuis le fichier HTML).

`<body></body>` est la balise « mère » de votre code HTML, rien n'est au-dessus mis à part la balise `<html></html>`. Lorsque vous rajoutez une image de fond dans body, cela permet donc de rajouter un fond à votre page. Par conséquent, lorsque vous aller ajouter des conteneurs contenant du texte par exemple, il y aura toujours le fond d'écran derrière. En effet, les propriétés appliquées sur un élément parent s'appliquent par la même occasion sur les éléments enfants. Pour avoir un texte plus lisible, vous serez obligé d'« écraser » cette propriété dans le fils en ajoutant une couleur de fond à cet élément.

METTRE DES EXEMPLES SCREENSHOTS !

Le fond de page peut être géré par plusieurs propriétés que vous devez connaître :

- `background-repeat` : la répétition de l'image de fond. En effet, l'image ne sera sûrement pas de la bonne taille et elle peut donc se répéter plusieurs fois pour couvrir l'intégralité du

fond d'écran. De plus, par défaut, lorsque vous scrollez l'image descend également. Elle peut donc se répéter pour ne pas laisser un blanc.

Les valeurs possibles pour cette propriété sont les suivantes :

- **repeat** : le fond est répété sous la forme d'une mosaïque (choisi par défaut).
- **no-repeat** : le fond n'est pas répété. Il y a donc une possibilité éventuelle d'avoir des espaces sans fond.
- **repeat-x** : le fond est répété sur une ligne, horizontalement.
- **repeat-y** : le fond est répété sur une colonne, verticalement.

- **background-attachement** : la fixation ou le défilement du fond d'écran.

Les valeurs possibles pour cette propriété sont les suivantes :

- **scroll** : le fond défile lors du scroll de l'écran.
- **fixed** : le fond reste figé lors du scroll de l'écran, seul le reste défile.

- **background-position** : la position de l'image de fond.

Les valeurs possibles pour cette propriété sont les suivantes :

- **top** : en haut
- **right** : à droite
- **bottom** : en bas
- **left** : à gauche
- **center** : au centre

On peut également les combiner entre eux, en voilà des exemple :

- **top center** : en haut au centre
- **bottom right** : en bas à droite

- Il est également possible d'utiliser des unités en pixel, en pourcentage, etc, à partir d'une origine donnée (par défaut le coin supérieur gauche de la page web) :

- **100px 10px** : placé à 100 pixels de la gauche et 10 pixels du haut de l'écran
- **2cm 3cm** : placé à 2 cm de la gauche et 3 cm du haut de l'écran
- **50% 6%** : placé à 50% de la longueur de l'écran par rapport à la gauche et à 6% de la hauteur de l'écran par rapport au haut

Toutes ses propriétés sont cumulables en utilisant la propriété background :

```
body {  
    background: url("img/background.png") fixed no-repeat bottom right;  
}
```

Dans cet exemple, j'ai appliqué une image de fond à la page web car c'est ce que l'on cherche à faire dans 90% des cas, mais gardez à l'esprit que l'on peut rajouter un fond à n'importe quelle balise HTML. Vous pouvez essayer à vos heures perdues c'est assez amusant.

L'opacité de notre fond

L'opacité peut se gérer de deux manières différentes.

La première consiste à utiliser la propriété **opacity**. Cette dernière prend pour valeur un nombre flottant (nombre à virgule) compris entre 0 et 1. Le chiffre 0 correspond à une opacité nulle, l'élément est donc totalement transparent, invisible en somme. Le chiffre 1, quant à lui réfère à une opacité totale, tout ce qui est derrière lui ne sera pas visible. En général, on cherche une valeur d'opacité intermédiaire pour créer un léger effet de transparence.

CODE CSS

La deuxième solution est d'utiliser la notation RGBa. C'est exactement le même principe que le RGB, sauf que l'on a un quatrième paramètre à régler qui est l'opacité. On le règle exactement de la même manière que la propriété `opacity`, c'est-à-dire avec un chiffre à virgule en 0 et 1.

CODE CSS

Notez bien que pour les nombres à virgule, nous n'utilisons pas la virgule : « , » mais bien le point : « . ». Cela fait partie des erreurs fréquemment rencontrées chez les débutants et c'est compréhensible.

La mise en forme du texte

La taille

La taille du texte se gère avec la propriété `font-size`.

Il existe deux types de tailles, la **taille absolue** et la **taille relative**. Retenez bien cette notion d'absolu et de relatif car c'est un concept qui revient souvent en CSS.

La taille absolue donne des unités de mesures qui sont inscrites dans le marbre et ne changent jamais, comme le pixel (px) ou le centimètre (cm). On les utilisera le moins possible car justement elles ne s'adaptent pas, et avec l'avènement des smartphones et autres tablettes, on préférera des unités de mesure plus souples qui s'adapteront mieux aux différentes tailles d'écran.

La taille relative est une mesure plus subtile car elle ne fait pas appel à des unités précises. Les valeurs possibles sont les suivantes :

- `xx-small` : tout petit
- `x-small` : vraiment petit
- `small` : petit
- `medium` : normal
- `large` : grand
- `x-large` : vraiment grand
- `xx-large` : énorme

L'autre alternative, celle que je vous recommande, est d'utiliser la mesure en em. Une taille normale est de 1 em. Ainsi si l'on veut écrire un petit peu plus petit qu'à la normale on met une mesure entre 0 et 1 em et si l'on veut grossir le texte on met une mesure supérieure à 1. Cela s'adapte mieux à l'écran puisque l'on ne met pas une taille « brute » mais une taille relative.

Les effets stylisés sur du texte

Mettre en gras

Pour mettre en gras, je vous rappelle une nouvelle fois que l'on n'utilise pas la balise ``. Cette dernière permet de montrer au navigateur web qu'un mot est important, rien de plus ! C'est le navigateur web qui, par défaut, le met en gras.

Pour mettre du texte en gras, on utilise en revanche la propriété CSS `font-weight`. Cette dernière possède deux valeurs possibles :

- `bold` : met le texte en gras.
- `normal` : ne fait rien (choisi par défaut sauf pour les balises ``).

Mais à quoi sert la deuxième valeur puisque le texte n'est pas en gras par défaut ?

Vous oubliez un petit détail, le texte n'est pas en gras par défaut pour le contenu de la balise ****. En appliquant la valeur **normal** à la propriété **font-weight** pour la balise ****, on peut donc lui retirer la mise en gras qui n'est pas forcément souhaitée.

La mise en italique

Même rappel avec cette fois-ci cependant la balise ****. Elle n'est pas faite pour mettre en italique mais pour mettre en avant un texte ou une partie de ce dernier. Par exemple, on veut pouvoir appliquer un style à toutes les citations utilisées sur notre site qui consiste entre autres à mettre les citations en italique, et ce, même si toutes les citations ne sont pas forcément importantes.

Pour ce faire, on va utiliser la propriété CSS **font-style** qui possède trois valeurs possibles :

- **italic** : met le texte en italique.
- **oblique** : met le texte sous une forme penchée, légèrement différente de l'italique.
- **normal** : ne met aucun effet de texte penché (choisi par défaut sauf pour la balise ****).

Comme vous vous en doutez, la valeur **normal** est utilisée pour annuler une mise en italique, typiquement sur les balises HTML ****.

La décoration du texte

Le reste des effets possibles est considéré par le CSS comme de la « décoration de texte ». Ainsi, ils sont regroupés sous la propriété **text-decoration** dont voici le spectre des valeurs possibles :

- **underline** : souligne le texte.
- **overline** : met une barre au-dessus du texte.
- **line-through** : barre le texte.
- **blink** : fait clignoter le texte. C'est la fonctionnalité la plus marrante de toutes ! Enfin, jusqu'à ce que vous soyez pris de violents maux de tête... En réalité, les effets clignotants sans souvent évités sur les sites internet. En plus, le clignotement n'est pas reconnu par tous les navigateurs web.
- **none** : ne fait rien (choisi par défaut).

La police

Pour changer la police du texte, cela se passe du côté de la propriété **font-family**. Les polices connues de la plupart des navigateurs sont :

- Arial
- Arial black
- Comic Sans MS
- Courier New
- Georgia
- Impact
- Sans-Serif
- Serif
- Times New Roman
- Trebuchet MS
- Verdana

Cependant, il est possible d'afficher d'autres polices de textes plus originales ou qui collent mieux à l'univers de votre site. Néanmoins, il y a de fortes chances pour que cette police ne fasse pas partie de la banque de police dont dispose l'utilisateur de votre site internet. Il faut alors la lui faire télécharger automatique. Pour ce faire, on utilise `@font-face`.

PASSAGE CHIANT A FINIR DEMAIN

L'alignement

Pour régler l'alignement de nos textes, on utilise la propriété `text-align` qui prend les valeurs suivantes :

- `left` : aligne le texte à gauche mais pas à droite (choisi par défaut).
- `right` : aligne le texte à droite mais pas à gauche.
- `center` : aligne le texte au centre.
- `justify` : aligne le texte à gauche et à droite.

Le positionnement flottant

Il est possible de « faire flotter » une image dans du texte, de telle manière que le texte englobe l'image en question.

Pour cela, rien de plus facile, il existe la propriété `float` qui peut prendre l'une des deux valeurs suivantes :

- `left` : l'élément flotte à gauche du texte.
- `right` : l'élément flotte à droite du texte.

Cependant, parfois l'image est trop haute et le texte que l'on a mis à côté est terminé. On aimerait alors que la suite du texte se place à nouveau en dessous de l'image en question.

C'est pour cela qu'a été créé la propriété `clear` qui peut prendre trois valeurs :

- `left` : replace le texte en dessous de l'image lorsque sa propriété `float` est à `left`.
- `right` : replace le texte en dessous de l'image lorsque sa propriété `float` est à `right`.
- `both` : replace le texte en dessous de l'image lorsque sa propriété `float` est à `left` ou à `right`.

On fait généralement un `clear : both`; car il est très rare d'arriver à un point de détail tel qu'on doit distinguer si le flottant est à droite ou à gauche.

Bordure et ombre

Les tableaux suite et fin