Les cahiers d'Exercices en Programmation : Le langage CSS

Niveau 2

Apprenez et entraînez vos acquis

- De très nombreux exercices à réaliser par vous-même accompagnés des solutions

Professeur : Bart Kimps

AVANT-PROPOS

Ce livre est un cahier d'exercices : il vous propose des énoncés d'exercices et leurs corrigés. Vous allez apprendre le logiciel en vous entrainant à travers des exercices regroupés par thème.

Chaque énoncé vous présente l'exercice à réaliser. Vous trouverez dans ce cahier le corrigé de chaque exercice. Certaines explications peuvent-être présentes.

METHODOLOGIE

Lors de la réalisation des exercices, vous pourrez remédier à certain problème à l'aide des corrections situé dans le cahier.

Après avoir réalisé tous les exercices de chaque chapitre vous allez pouvoir vérifier les compétences acquises à l'aide des exercices synthèses, des questions Quizz, etc.

Des **légendes ou recommandations** peuvent être présentes dans certains exercices. Celles-ci vous aideront dans vos recherches. Elles ne doivent pas être reproduites dans votre travail.

Chaque point de matière acquis dans un exercice peut être utilisé dans des exercices suivants sans explication.

Table des matières

La mise en page du site

Chapitre 1 : Structurer sa page

Il nous manque encore quelques connaissances nécessaires pour faire la mise en page.

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

Dans ce chapitre, nous allons nous intéresser aux nouvelles balises HTML dédiées à la structuration du site. Ces balises ont été introduites par HTML5 (elles n'existaient pas avant) et vont nous permettre de dire : « Ceci est mon en-tête », « Ceci est mon menu de navigation », etc.

Pour le moment, nous n'allons pas encore faire de mise en page. Nous allons en fait préparer notre document HTML pour pouvoir découvrir la mise en page dans les prochains chapitres.

Au point suivant, tu vas essayer d'utiliser des balises que tu vas découvrir pour structurer ta page web.

Les balises structurantes de l'HTML5

Tu vas créer cette page Web une étape à la fois.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Mon site Web pour initier l'informatique à tout le monde</title>
    </head>

<body>
```

Pour l'instant tu connais.

L'exercice terminé se retrouve en fin de paragraphe.

Tu vas maintenant passer au corps de la page Web.

<Header> : l'en-tête

La plupart des sites web possèdent en général un en-tête, appelé header en anglais. On y trouve le plus souvent un logo, une bannière, le slogan de votre site…

```
        <header>
            <h1>Kimps Bart</h1>
            <h2>Différents cours</h2>
        </header>
```

<footer> : le pied de page

À l'inverse de l'en-tête, le pied de page se trouve en général tout en bas du document. On y trouve des informations comme des liens de contact, le nom de l'auteur, les mentions légales, etc.

Passe pour cela beaucoup de lignes "ENTER" afin d'arriver comme ci-dessous

```
        <footer>
            <p>Copyright Bart - Tous droits réservés<br />
            <a href="#">Me contacter !</a></p>
        </footer>
    </body>
</html>
```

<nav> : principaux liens de navigation

La balise <nav> doit regrouper tous les principaux liens de navigation du site. On y place par exemple le menu principal du site.

Généralement, le menu est réalisé sous forme de liste à puces à l'intérieur de la balise <nav> :

```
14      <nav>
15          <ul>
16              <li><a href="#">Accueil</a></li>
17              <li><a href="#">Débutant</a></li>
18              <li><a href="#">Confirmé</a></li>
19          </ul>
20      </nav>
```

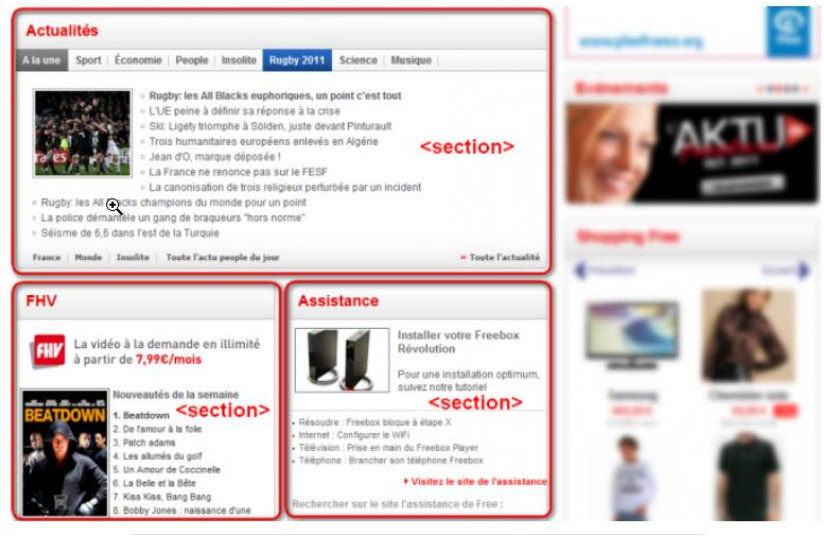
Les liens sont volontairement factices (d'où la présence d'un simple #), ils n'amènent donc nulle part (eh, c'est juste un exercice découverte) !

<section> : une section de page

La balise <section> sert à regrouper des contenus en fonction de leur thématique. Elle englobe généralement une portion du contenu au centre de la page.

```
22      <section>
23          <aside>
24              <h1>À propos de l'auteur</h1>
25              <p>C'est moi, le prof ! Je suis né un jour, il y a déjà des années</p>
26          </aside>
27
28          <article>
29              <h1>J'aime apprendre et faire apprendre</h1>
30              <p>A découvrir</p>
31          </article>
32      </section>
```

Voici un exemple plus explicite :



Remarque : Chaque section peut avoir son titre de niveau 1 (<h1>), de même que l'en-tête peut contenir un titre <h1> lui aussi. Chacun de ces blocs étant indépendant des autres, on peut donc retrouver plusieurs titres <h1> dans le code de la page web. On a ainsi « Le titre <h1> du <header> », « Le titre <h1> de cette <section> », etc.

<aside> : informations complémentaires

On retrouve la balise <aside> dans la balise <section>, pourquoi ?

La balise <aside> est conçue pour contenir des informations complémentaires au document que l'on visualise. Ces informations sont généralement placées sur le côté (bien que ce ne soit pas une obligation).

Voici un exemple plus explicite :

Rien à insérer de plus dans ta page Web car ce point a déjà été réalisé précédemment.

<article> : un article indépendant

La balise <article> sert à englober une portion généralement autonome de la page. C'est une partie de la page qui pourrait ainsi être reprise sur un autre site. C'est le cas par exemple des actualités (articles de journaux ou de blogs).

Voici un exemple plus explicite :

Rien à insérer de plus dans ta page Web car ce point a déjà été réalisé précédemment.

Exercice complet :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Mon site Web pour initier l'informatique à tout le monde</title>
    </head>

<body>
        <header>
            <h1>Kimps Bart</h1>
            <h2>Différents cours</h2>
        </header>

        <nav>
            <ul>
                <li><a href="#">Accueil</a></li>
                <li><a href="#">Débutant</a></li>
                <li><a href="#">Confirmé</a></li>
            </ul>
        </nav>

        <section>
            <aside>
                <h1>À propos de l'auteur</h1>
                <p>C'est moi, le prof ! Je suis né un jour, il y a déjà des années</p>
            </aside>

            <article>
                <h1>J'aime apprendre et faire apprendre</h1>
                <p>A découvrir</p>
            </article>
        </section>

        <footer>
            <p>Copyright Bart - Tous droits réservés<br />
            <a href="#">Me contacter !</a></p>
        </footer>
    </body>
</html>
```

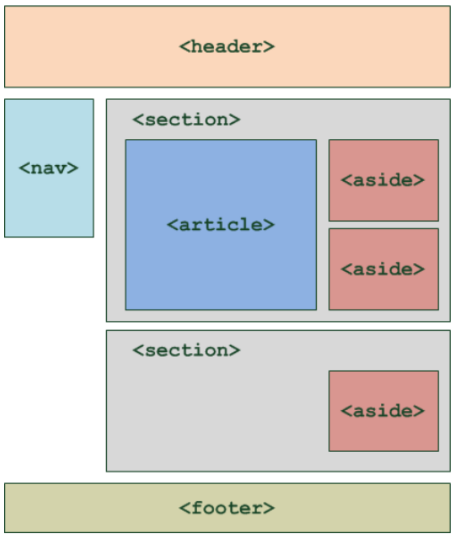
Pour l'instant la page Web ne ressemble à rien mais ne t'inquiète pas nous allons modifier cela dans les chapitres suivants.

Notamment intégrer du CSS.

Résumé

Ouf, cela fait beaucoup de nouvelles balises à retenir.
Voici un schéma exemple pour t'aider à retenir leur rôle !

Mais ton imagination peut aussi créer une autre organisation.



- Plusieurs balises ont été introduites avec HTML5 pour délimiter les différentes zones qui constituent la page web :
  - <header> : en-tête ;
  - <footer> : pied de page ;
  - <nav> : liens principaux de navigation ;
  - <section> : section de page ;
  - <aside> : informations complémentaires ;
  - <article> : article indépendant.
- Ces balises peuvent être imbriquées les unes dans les autres. Ainsi, une section peut avoir son propre en-tête.
- Ces balises ne s'occupent pas de la mise en page. Elles servent seulement à indiquer à l'ordinateur le sens du texte qu'elles contiennent. On pourrait très bien placer l'en-tête en bas de la page si on le souhaite.

Chapitre 2 : Le modèle des boîtes

Une page web peut être vue comme une succession et un empilement de boîtes, qu'on appelle « blocs ». La plupart des éléments vus au chapitre précédent sont des blocs : <header>, <article>, <nav>…

Mais nous connaissions déjà d'autres blocs : les paragraphes , les titres <h1>…

Nous allons apprendre à manipuler ces blocs comme de véritables boîtes. Nous allons leur donner des dimensions, les agencer en jouant sur leurs marges, mais aussi apprendre à gérer leur contenu… pour éviter que le texte ne dépasse de ces blocs !

Ce sont des notions fondamentales dont nous allons avoir besoin pour mettre en page notre site web… Cela se complique !

Les balises de type block et inline

En HTML, la plupart des balises peuvent se ranger dans l'une ou l'autre de deux catégories :

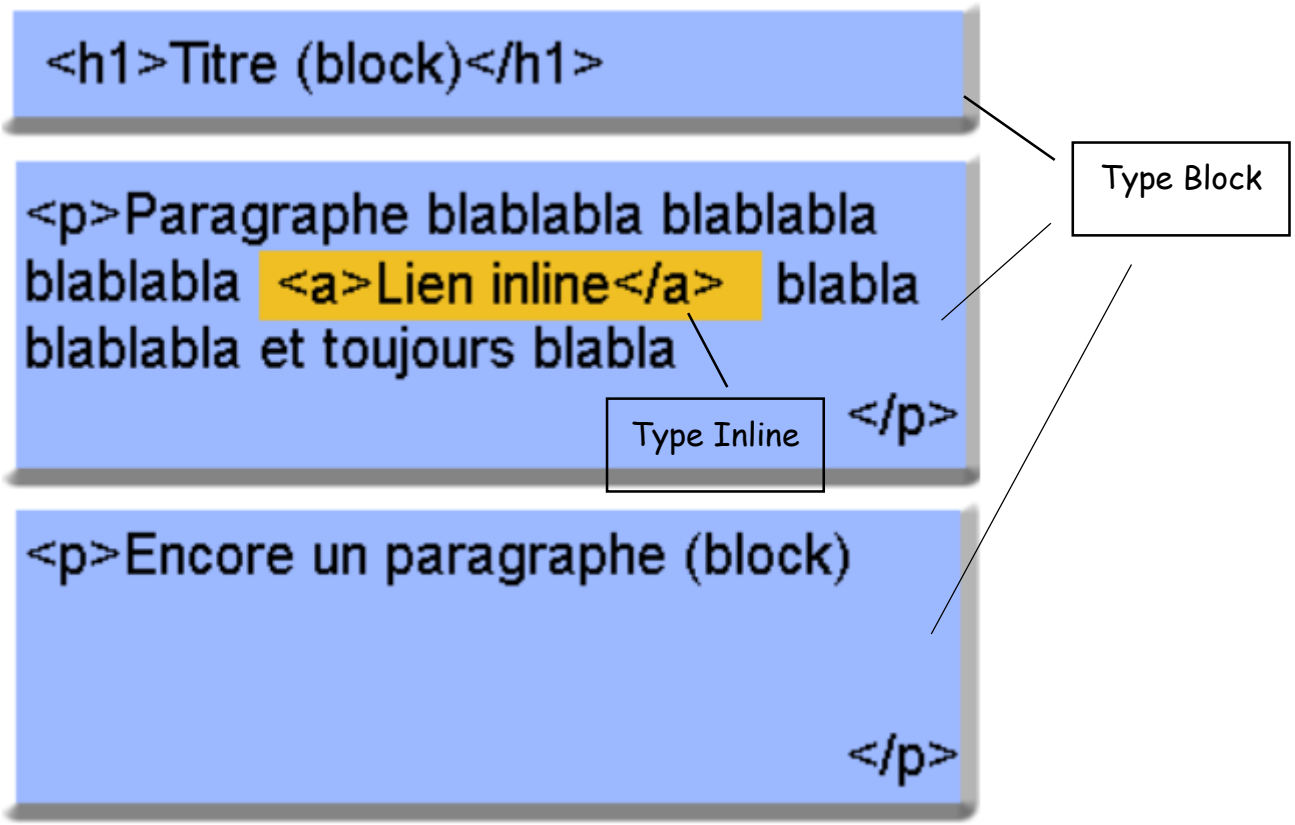
- Les balises inline : c'est le cas par exemple des liens <a> .
- Les balises block : c'est le cas par exemple des paragraphes .

Mais comment reconnaitre une balise inline à une balise block ?

C'est en fait assez facile :

- block : une balise de type block sur votre page web crée automatiquement un retour à la ligne avant et après. Il suffit d'imaginer un bloc. Votre page web sera en fait constituée d'une série de blocs les uns à la suite des autres. Mais vous verrez qu'en plus, il est possible de mettre un bloc à l'intérieur d'un autre, ce qui va augmenter considérablement nos possibilités pour créer le design de notre site !
- inline : une balise de type inline se trouve obligatoirement à l'intérieur d'une balise block. Une balise inline ne crée pas de retour à la ligne, le texte qui se trouve à l'intérieur s'écrit donc à la suite du texte précédent, sur la même ligne (c'est pour cela que l'on parle de balise « en ligne »).

Voici un petit schéma à la figure suivante :



Comme on peut le voir, les blocs sont les uns en-dessous des autres. On peut aussi les imbriquer les uns à l'intérieur des autres (blocs `<section>` contiennent par exemple des blocs `<aside>` !).
La balise inline `<a> </a>`, elle, se trouve à l'intérieur d'une balise block et le texte vient s'insérer sur la même ligne.

Autres exemples :

Balises block	Balises inline
<p>	
<footer>	
<h1>	<mark>
<h2>	<a>
<article>	<img />
...	...

Les balises universelles

Ce sont des balises qui n'ont aucun sens particulier (contrairement à <p> qui veut dire « paragraphe », « important », etc.).
Le principal intérêt de ces balises est que l'on peut leur appliquer une class (ou une id) pour le CSS quand aucune autre balise ne convient.

Il existe deux balises génériques et, comme par hasard, la seule différence entre les deux est que l'une d'elle est inline et l'autre est block :

-   (**inline**) ;
-   (**block**).

Les dimensions

Un bloc a des dimensions précises. Il possède une largeur et une hauteur. On dispose de deux propriétés CSS :

- width: c'est la largeur du bloc. À exprimer en pixels (px) ou en pourcentage (%).
- height: c'est la hauteur du bloc. À exprimer soit en pixels (px), soit en pourcentage (%).

Un bloc prend par défaut 100% de la largeur disponible. On peut le vérifier en appliquant à nos blocs des bordures ou une couleur de fond.

Exercices :

Tu vas créer et joindre à ton site Web un fichier CSS nommé Mon Site.css

Fichier HTML à modifier (réaliser un lien) :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Mon site Web pour initier l'informatique à tout le monde</title>
        <link rel="stylesheet" href="Mon Site.css" />
    </head>
```

Fichier CSS :

Afin de modifier la largeur des paragraphes. Le CSS suivant dit : « Je veux que tous mes paragraphes aient une largeur de 50% » + autres attributs.

```
p{
    background-color: grey;
    background: underline;
    width: 50%;

    font-size: 20px;
    font-family: "Comic sans MS";
}
```

Regarde le résultat.

Minimum et maximum

On peut demander à ce qu'un bloc ait des dimensions minimales et maximales. Pratique car cela nous permet de définir des dimensions « limites » pour que notre site s'adapte aux différentes résolutions d'écran de nos visiteurs :

- min-width: largeur minimale ;
- min-height: hauteur minimale ;
- max-width: largeur maximale ;
- max-height: hauteur maximale.

Par exemple, on peut demander à ce que les paragraphes occupent 50% de la largeur et exiger qu'ils fassent au moins 200 pixels de large dans tous les cas :

```
p{
    background-color: grey;
    background: underline;
    width: 50%;
    min-width: 200px;

    font-size: 20px;
    font-family: "Comic sans MS";
}
```

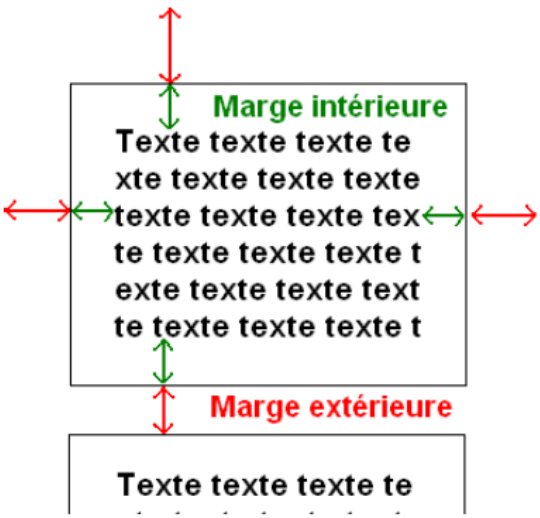
Observez le résultat en modifiant la largeur de la fenêtre de votre navigateur. Vous allez voir que, si celle-ci est trop petite, le paragraphe se force à occuper au moins 200 pixels de largeur.

Les marges

Il faut savoir que tous les blocs possèdent des marges. Il existe *deux types de marges* :

- les marges intérieures ;
- les marges extérieures.

Regarde ce schéma :



En CSS, on peut modifier la taille des marges avec les deux propriétés suivantes :

- padding: indique la taille de la marge intérieure. À exprimer en général en pixels (px).
- margin: indique la taille de la marge extérieure. Là encore, on utilise le plus souvent des pixels.

```css
p{
    background-color: grey;
    background: underline;
    width: 50%;
    min-width: 200px;

    font-size: 20px;
    font-family: "Comic sans MS";

    padding: 20px; /*Marge intérieure*/
    margin: 50px; /*Marge extérieure*/
}
```

Centrer des blocs

Il est tout à fait possible de centrer des blocs. C'est même très pratique pour réaliser un design centré quand on ne connaît pas la résolution du visiteur.

Pour centrer, il faut respecter les règles suivantes :

- donnez une largeur au bloc (avec la propriété width) ;
- indiquez que vous voulez des marges extérieures automatiques, comme ceci : margin: auto;

Appliquons ce point à la balise structurante <header> qui est l'entête :

```css
header
{
    background-color: grey;
    background: underline;
    width: 50%;
    min-width: 400px;
    font-size: 20px;
    font-family: "Comic sans MS";
    width: 50%;
    margin: auto;
    border: 1px solid black;
    text-align: justify;
    padding: 12px;
    margin-bottom: 20px;
    text-align: center;
}
```

Quand ça dépasse… : Overflow

Lorsqu'on commence à définir des dimensions précises pour nos blocs, comme on vient de le faire, il arrive qu'ils deviennent trop petits pour le texte qu'ils contiennent.

Les propriétés CSS que tu vas découvrir ici ont été créées pour contrôler les dépassements… et décider quoi faire si jamais cela devait arriver.

Insère du texte dans ton fichier HTML comme ci-dessous :

```
<body>
        <header>
            <h1>Kimps Bart</h1>
            <h2>Différents cours</h2>
        </header>

        <nav>
        <p>
        C'est ici que vous allez choisir ce que vous souhaitez apprendre.
        Nous vous proposons actuellement des cours en ligne sans bouger de chez vous
        pour apprendre à utiliser votre ordinateur Windows, un tratiement de texte,
        un tableur, ...
        N'hésitez pas à en parler autour de vous, nos cours sont gratuits et ouverts à tous !
        </p>
            <ul>
                <li><a href="#">Accueil</a></li>
                <li><a href="#">Débutant</a></li>
                <li><a href="#">Confirmé</a></li>
            </ul>
        </nav>
```

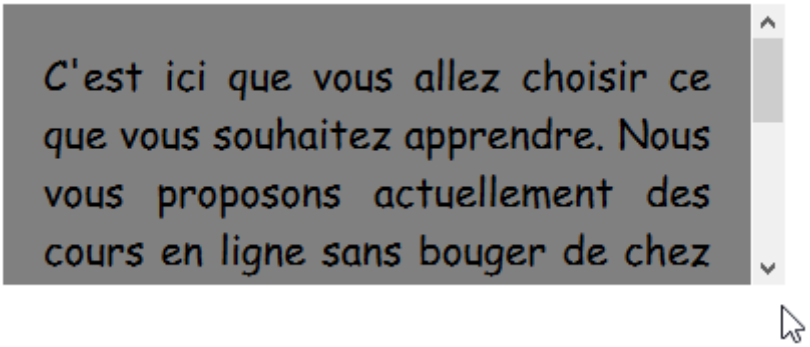
Modifie et ajoute certaines lignes dans ton fichier CSS :

```
p{
    background-color: grey;
    background: underline;
    width: 350px;
    height : 100px;
    text-align : justify;

    font-size: 20px;
    font-family: "Comic sans MS";

    padding: 20px; /*Marge intérieure*/
    margin: 50px; /*Marge extérieure*/

    overflow: visible;
}
```

Regarde le résultat et maintenant modifie la valeur de la dernière ligne :

```
overflow: auto;
```

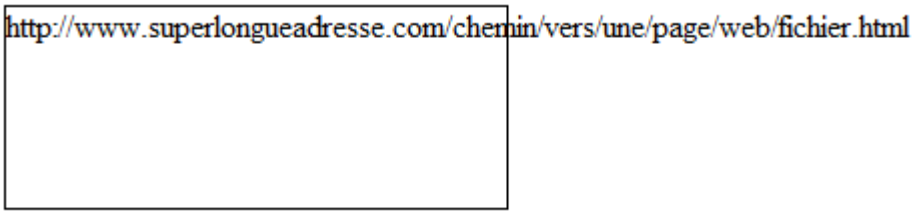
Voici les modifications apportées :



Couper les textes trop larges avec Word-wrap

Si vous devez placer un mot très long dans un bloc, qui ne tient pas dans la largeur, vous allez adorer word-wrap. Cette propriété permet de forcer la césure des très longs mots (généralement des adresses un peu longues).

La figure suivante représente ce que l'on peut avoir quand on écrit une URL un peu longue dans un bloc.



Le texte déborde en largeur

L'ordinateur ne sait pas « couper » l'adresse car il n'y a ni espace, ni tiret. Il ne sait pas faire la césure.

Avec le code suivant, la césure sera forcée dès que le texte risque de dépasser (figure suivante).

```
p{
    background-color: grey;
    background: underline;
    width: 350px;
    height : 100px;
    text-align : justify;

    font-size: 20px;
    font-family: "Comic sans MS";

    padding: 20px; /*Marge intérieure*/
    margin: 50px; /*Marge extérieure*/

    overflow: auto;

    word-wrap: break-word;
}
```

En résumé

- On distingue deux principaux types de balises en HTML :
    - Le type block (<p>,<h1>…) : ces balises créent un retour à la ligne et occupent par défaut toute la largeur disponible. Elles se suivent de haut en bas.
    - Le type inline (<a>, …) : ces balises délimitent du texte au milieu d'une ligne. Elles se suivent de gauche à droite.
- On peut modifier la taille d'une balise de type block avec les propriétés CSS width (largeur) et height (hauteur).
- On peut définir des minima et maxima autorisés pour la largeur et la hauteur : min-width, max-width, min-height, max-height.
- Les éléments de la page disposent chacun de marges intérieures (padding) et extérieures (margin).
- S'il y a trop de texte à l'intérieur d'un bloc de dimensions fixes, il y a un risque de débordement. Dans ce cas, il peut être judicieux de rajouter des barres de défilement avec la propriété overflow ou de forcer la césure avec word-wrap.

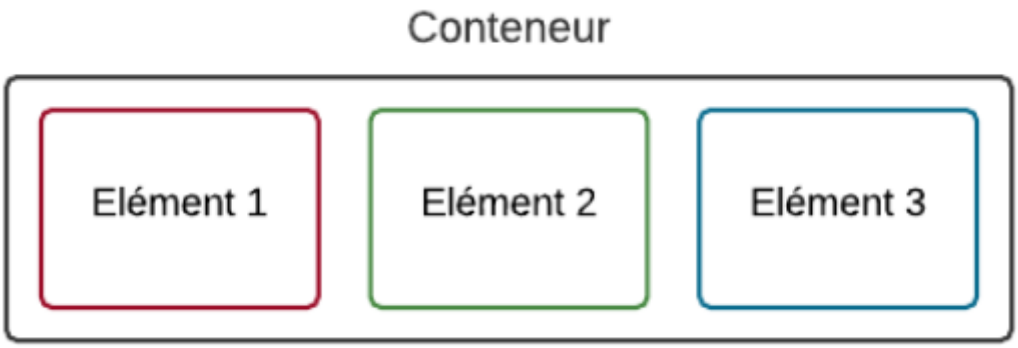
Chapitre 3 : La mise en page avec Flexbox

Il y a plusieurs façons de mettre en page un site. Au fil du temps, plusieurs techniques ont existé mais nous nous intéresserons à la dernière qui est la meilleure : **Flexbox** ! Elle permet toutes les folies (ou presque 😉) et c'est celle que je vous recommande d'utiliser si vous en avez la possibilité, lorsque vous créez un nouveau site. Flexbox est désormais reconnu par tous les navigateurs récents !

Un conteneur, des éléments

Le principe de la mise en page avec Flexbox est simple : vous définissez un conteneur, et à l'intérieur vous placez plusieurs éléments. Imaginez un carton dans lequel vous rangez plusieurs objets : c'est le principe.

Sur une même page web, vous pouvez sans problème avoir plusieurs conteneurs.



Pour ce chapitre crée un nouveau fichier HTML nommé Flexbox et un fichier CSS nommé Style Flexbox.

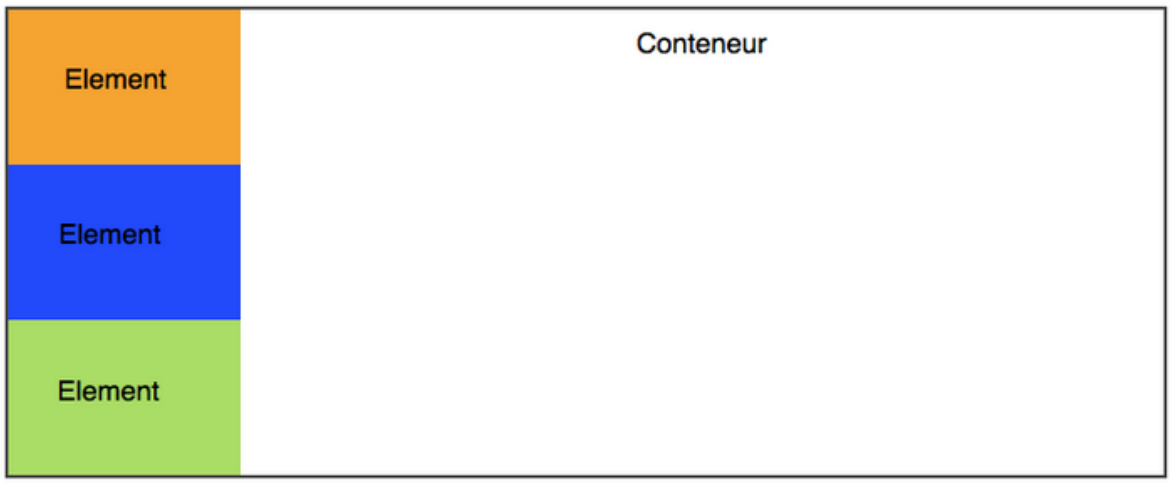
Dans le fichier HTML :

```html
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="utf-8" />
        <title>Flexbox</title>
        <link rel="stylesheet" href="Style Flexbox.css" />
    </head>

<body>
    <div id="conteneur">
        <div class="element">Elément 1</div>
        <div class="element">Elément 2</div>
        <div class="element">Elément 3</div>
    </div>

</body>
```

Dans le fichier CSS:

```css
#conteneur
{
    border: 2px solid #444;
}

.element
{
    width: 150px;
    height: 100px;
}

.element:nth-child(1)
{
    background-color: orange;
}

.element:nth-child(2)
{
    background-color: blue;
}

.element:nth-child(3)
{
    background-color: #ad6;
}
```

Exécute ton fichier HTML. Les blocs se positionnent ici les uns en dessous des autres.



Les directions et le sens

Dans le fichier CSS, modifie le bloc #conteneur en ajoutant cette ligne :

```
#conteneur
{
    Display: flex;
    border: 2px solid #444;
}
```

Les blocs se placeront maintenant les uns à la suite des autres sur une même ligne.

Pour changer leur alignement, on va utiliser justify-content, qui peut prendre ces valeurs :

- flex-start : alignés au début (par défaut)
- flex-end : alignés à la fin
- center : alignés au centre
- space-between : les éléments sont étirés sur tout l'axe (il y a de l'espace entre eux)
- space-around : idem, les éléments sont étirés sur tout l'axe, mais ils laissent aussi de l'espace sur les extrémités

Voici un exemple :

```
#conteneur
{
    Display: flex;
    border: 2px solid #444;
    justify-content: flex-end;
}
```

À toi de tester les autres valeurs.

La direction et le sens peuvent être modifiés à l'aide de cette ligne :

```
#conteneur
{
    Display: flex;
    border: 2px solid #444;
    justify-content: flex-end;
    flex-direction: column;
}
```

Et si on inversait tout.

```
#conteneur
{
    Display: flex;
    border: 2px solid #444;
    justify-content: flex-end;
    flex-direction: column-reverse;
}
```

Voici d'autres options :

- row : organisés sur une ligne (par défaut)
- column : organisés sur une colonne
- row-reverse : organisés sur une ligne, mais en ordre inversé
- column-reverse : organisés sur une colonne, mais en ordre inversé

Le retour à la ligne

Par défaut, les blocs essaient de rester sur la même ligne s'ils n'ont pas la place (ce qui peut provoquer des bugs de design parfois). Les blocs peuvent aller à la ligne lorsqu'ils n'ont plus la place avec flex-wrap qui peut prendre ces valeurs :

- nowrap : pas de retour à la ligne (par défaut)
- wrap : les éléments vont à la ligne lorsqu'il n'y a plus la place
- wrap-reverse : les éléments vont à la ligne lorsqu'il n'y a plus la place en sens inverse

Et encore, et encore …

En résumé

- Il existe plusieurs techniques pour positionner les blocs sur la page. Flexbox est la technique la plus récente et de loin la plus puissante, que je vous recommande d'utiliser.
- Le principe de Flexbox est d'avoir un conteneur, avec plusieurs éléments à l'intérieur. Avec display: flex; sur le conteneur, les éléments à l'intérieur sont agencés en mode Flexbox (horizontalement par défaut).
- Flexbox peut gérer toutes les directions. Avec flex-direction, on peut indiquer si les éléments sont agencés horizontalement (par défaut) ou verticalement. Cela définit ce qu'on appelle l'axe principal.
- L'alignement des éléments se fait sur l'axe principal avec justify-content , et sur l'axe secondaire avec align-items.
- Avec flex-wrap, on peut autoriser les éléments à revenir à la ligne s'ils n'ont plus d'espace.
- S'il y a plusieurs lignes, on peut indiquer comment les lignes doivent se répartir entre elles avec align-content.
- Chaque élément peut être ré agencé en CSS avec order (pas besoin de toucher au code HTML !).
- Avec la super-propriété flex, on peut autoriser nos éléments à occuper plus ou moins d'espace restant.
- Flexbox, c'est cool.

Chapitre 4 : Autres techniques de mise en page

Reprenons l'exercice HTML et CSS mon site Web

Mettez les lignes de votre fichier CSS actuel en commentaire /* */

Modifie le fichier HTML :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Mon site Web pour initier l'informatique à tout le monde</title>
        <link rel="stylesheet" href="Mon Site.css" />
    </head>

<body>
        <header>
            <h1>Kimps Bart</h1>
            <h2>Différents cours</h2>
        </header>
        <p div class="par">
        C'est ici que vous allez choisir ce que vous souhaitez apprendre.
        Nous vous proposons actuellement des cours en ligne sans bouger de chez vous
        pour apprendre à utiliser votre ordinateur Windows, un tratiement de texte,
        un tableur, ...
        N'hésitez pas à en parler autour de vous, nos cours sont gratuits et ouverts à tous !
        </p>
        <nav>
            <ul>
                <li><a href="#">Accueil</a></li>
                <li><a href="#">Débutant</a></li>
                <li><a href="#">Confirmé</a></li>
            </ul>
        </nav>
        <section>
            <aside>
                <h1>À propos de l'auteur</h1>
                <p>C'est moi, le prof ! Je suis né un jour, il y a déjà des années</p>
            </aside>

            <article>
                <h1>J'aime apprendre et faire apprendre</h1>
                <p>A découvrir</p>
            </article>
        </section>
        <footer>
            <p>Copyright Bart - Tous droits réservés<br />
            <a href="#">Me contacter !</a></p>
        </footer>
    </body>
</html>
```

Ajoute ces lignes dans le fichier CSS.

Explications :

- Le menu restera à gauche de la page.
- Un bloc a été défini (.par) et des attributs ont été appliqués
- Les sections seront bordées d'une ligne bleue de 1px d'épaisseur

```css
nav
{
    float: left;
    width: 150px;
    border: 1px solid black;
}

section
{
    border: 1px solid blue;
}

.par
{
    text-align: center;
    background-color: grey;
    width: 800px;
    height : 120px;
    font-size: 15px;
    font-family: "Comic sans MS";
}
```

Il y a deux défauts (mis à part le fait que c'est encore bien moche) :

- Le texte du corps de la page touche la bordure du menu. Il manque une petite marge…
- Plus embêtant encore : la suite du texte passe… sous le menu !

On veut bien que le pied de page (« Copyright Zozor ») soit placé en bas sous le menu mais, par contre, on aimerait que tout le corps de page soit constitué d'un seul bloc placé à droite.

Pour résoudre ces deux problèmes d'un seul coup, il faut ajouter une marge extérieure à gauche de notre <section>, marge qui doit être par ailleurs supérieure à la largeur du menu. Si notre menu fait 150 px, nous allons par exemple donner une marge extérieure gauche de 170 px à notre section de page.

```
8   section
9   {
10      margin-left: 170px;
11      border: 1px solid blue;
12  }
```

Transformer les éléments avec display

Les lois du CSS vont être ici changées (brrr…). Il faut s'accrochers !

Il existe en CSS une propriété très puissante : `display`. Elle est capable de *transformer* n'importe quel élément de votre page d'un type vers un autre. Avec cette propriété magique, je peux par exemple imposer à mes liens (originellement de type inline) d'apparaître sous forme de blocs :

```
a
{
    display: block;
}
```

À ce moment-là, les liens vont se positionner les uns en-dessous des autres (comme des blocs normaux) et il devient possible de modifier leurs dimensions !

Voici quelques-unes des principales valeurs que peut prendre la propriété display en CSS (il y en a encore d'autres) :

Valeur	Exemples	Description
`inline`	`<a>`, `<em>`, `<span>`...	Eléments d'une ligne. Se placent les uns à côté des autres.
`block`	`<p>`, `<div>`, `<section>`...	Eléments en forme de blocs. Se placent les uns en-dessous des autres et peuvent être redimensionnés.
`inline-block`	`<select>`, `<input>`	Eléments positionnés les uns à côté des autres (comme les inlines) mais qui peuvent être redimensionnés (comme les blocs).
`none`	`<head>`	Eléments non affichés.

On peut donc décider de masquer complètement un élément de la page avec cette propriété. Par exemple, si je veux masquer les éléments qui ont la classe « secret », je vais écrire :

Le positionnement inline-block

Les manipulations que demande le positionnement flottant se révèlent parfois un peu délicates sur des sites complexes. Dès qu'il y a un peu plus qu'un simple menu à mettre en page, on risque d'avoir à recourir à des `clear: both;` qui complexifient rapidement le code de la page.

Une meilleure technique consiste à transformer les éléments en `inline-block` avec la propriété `display`.

Petits rappels sur les éléments de type `inline-block` :

- Ils se positionnent les uns à côté des autres; (placer un menu et un corps sur une page !).
- On peut leur donner des dimensions précises ;
- Je crois que c'est ce que l'on veut.

Nous allons transformer en `inline-block` les deux éléments que nous voulons placer côte à côte : le menu de navigation et la section du centre de la page.

Modifie ces 2 classes :

```
nav
{
    display: inline-block;
    float: left;
    width: 150px;
    border: 1px solid black;
}

section
{
    display: inline-block;
    margin-left: 170px;
    border: 1px solid blue;
}
```

Le fait d'avoir transformé les éléments en inline-block nous permet d'utiliser une nouvelle propriété, normalement réservée aux tableaux : vertical-align. Cette propriété permet de modifier l'alignement vertical des éléments. Voici quelques-unes des valeurs possibles pour cette propriété :

- baseline : aligne de la base de l'élément avec celle de l'élément parent (par défaut) ;
- top : aligne en haut ;
- middle : centre verticalement ;
- bottom : aligne en bas ;
- (valeur en px ou %) : aligne à une certaine distance de la ligne de base (baseline).

```
nav
{
    display: inline-block;
    float: left;
    width: 150px;
    border: 1px solid black;
    vertical-align: top;
}

section
{
    display: inline-block;
    margin-left: 170px;
    border: 1px solid blue;
    vertical-align: top;
}
```

Le menu est aligné avec le corps de la page.

Peut-être aucune modification n'a été visible. Mais grâce à ces deux lignes ajoutées les deux éléments seront toujours alignés avec le positionnement désiré.

Les positionnements absolu, fixe et relatif

Il existe d'autres techniques un peu particulières permettant de positionner avec précision des éléments sur la page :

- **Le positionnement absolu** : il nous permet de placer un élément n'importe où sur la page (en haut à gauche, en bas à droite, tout au centre, etc.).
- **Le positionnement fixe** : identique au positionnement absolu mais, cette fois, l'élément reste toujours visible, même si on descend plus bas dans la page. C'est un peu le même principe que `background-attachment: fixed;`
- **Le positionnement relatif** : permet de décaler l'élément par rapport à sa position normale.

Il faut faire son choix entre les trois modes de positionnement disponibles. Pour cela, on utilise la propriété CSS `position` à laquelle on donne une de ces valeurs :

- absolute : positionnement absolu ;
- fixed : positionnement fixe ;
- relative : positionnement relatif.

Nous allons étudier un à un chacun de ces positionnements.

Le positionnement absolu

Le positionnement absolu permet de placer un élément (réellement) n'importe où sur la page. Donc il faut aussi indiquer ou l'élément sera placé à l'aide de ces 4 propriétés CSS :

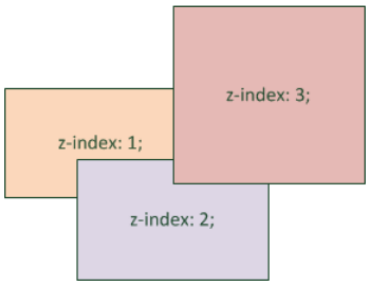
- left : position par rapport à la gauche de la page ;
- right : position par rapport à la droite de la page ;
- top : position par rapport au haut de la page ;
- bottom : position par rapport au bas de la page.

Voici un exemple. Le bloc `footer` sera placé en bas à droite de la page :

```
footer
{
    position: absolute;
    right: 0px;
    bottom: 0px;
}
```

Il est possible de placer des éléments l'un au-dessus de l'autre à l'aide de la propriété z-index=1,2,3,4, …

L'élément ayant la valeur de z-index la plus élevée sera placé par-dessus les autres.



Le positionnement fixe

Le principe est *exactement le même* que pour le positionnement absolu sauf que, cette fois, le bloc reste fixe à sa position, même si on descend plus bas dans la page.

On ne testera pas cette propriété qui je crois n'est pas trop difficile à comprendre (position: fixed;)

Le positionnement relatif

Plus délicat, le positionnement relatif peut vite devenir difficile à utiliser. Ce positionnement permet d'effectuer des « ajustements » : l'élément est décalé par rapport à sa position initiale.

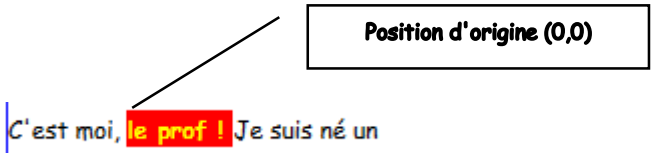
Prenons par exemple un texte important, situé entre deux balises . Pour commencer, je le mets sur fond rouge pour qu'on puisse mieux le repérer :

Fichier HTML :

```
<section>
    <aside>
        <h1>À propos de l'auteur</h1>
        <p>C'est moi, <strong>le prof ! </strong>Je suis né un jour, il y a déjà des années</p>
    </aside>

    <article>
        <h1>J'aime apprendre et faire apprendre</h1>
        <p>A découvrir</p>
    </article>
</section>
```

Fichier CSS :

```css
strong
{
    background-color: red; /* Fond rouge */
    color: yellow; /* Texte de couleur jaune */
}
```

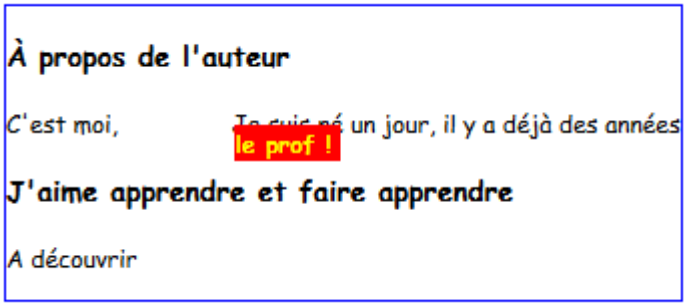


Donc si tu fais position: relative; et que tu appliques une des propriétés top, left, right ou bottom, le texte sur fond rouge va se déplacer par rapport à la position où il se trouve.

Prenons un exemple : je veux que mon texte se décale de 55 pixels vers la droite et de 10 pixels vers le bas. Je vais donc demander à ce qu'il soit décalé de 55 pixels par rapport au « bord gauche » et de 10 pixels par rapport au « bord haut » (lignes 6 à 8) :

```
strong
{
    background-color: red; /* Fond rouge */
    color: yellow; /* Texte de couleur jaune */

    position: relative;
    left: 55px;
    top: 10px;
}
```

Résultat :



En résumé

- La technique de mise en page la plus récente et la plus puissante est Flexbox. C'est celle que vous devriez utiliser si vous en avez la possibilité.
- D'autres techniques de mise en page restent utilisées, notamment sur des sites plus anciens : le positionnement flottant et le positionnement inline-block . Il est conseillé de les connaître.
- Le positionnement flottant (avec la propriété float) est l'un des plus utilisés à l'heure actuelle. Il permet par exemple de placer un menu à gauche ou à droite de la page. Néanmoins, cette propriété n'a pas été initialement conçue pour cela et il est préférable, si possible, d'éviter cette technique.
- Le positionnement inline-block consiste à affecter un type inline-block à nos éléments grâce à la propriété display. Ils se comporteront comme des inlines (placement de gauche à droite) mais pourront être redimensionnés comme des blocs (avec width et height). Cette technique est à préférer au positionnement flottant.
- Le positionnement absolu permet de placer un élément où l'on souhaite sur la page, au pixel près.
- Le positionnement fixe est identique au positionnement absolu mais l'élément restera toujours visible même si on descend plus bas dans la page.
- Le positionnement relatif permet de décaler un bloc par rapport à sa position normale.
- Un élément A positionné en absolu à l'intérieur d'un autre élément B (lui-même positionné en absolu, fixe ou relatif) se positionnera par rapport à l'élément B, et non par rapport au coin en haut à gauche de la page.

Chapitre 5 : Exercices

Tu vas effectuer la mise en page avec CSS. Encore un défi !

Il faut retrouver sur ta page Web les points suivants. Ceux-ci montreront que la matière à bien été acquise.

1. Une balise sémantique "nav" manque. Pouvez-vous l'ajouter au bon endroit ?
2. Retirer les puces de la liste à puces (à vous de trouver comment faire !)
3. Placer le header et le menu côte à côte. Créer une bordure autour de ces 2 éléments et ajouter une couleur de fond différente à chacun de ceux-ci.
4. Le plus gros point à réaliser (en CSS) :
    1. Afficher les paragraphes en justifié
    2. Définir une largeur et une hauteur bien spécifique aux paragraphes (550px, 120px)
    3. Centrer les paragraphes dans leur bloc
    4. Associer une couleur de fond identique pour les paragraphes
    5. Souligner les titres H1 des paragraphes en rouge (en CSS)
    6. Créer des espacements </br> afin d'éclaircir la page
    7. Choisir une police Comic Sans MS

Avant de commencer, tu dois créer un fichier HTML nommé : Index Leblog

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <title>Le blog trotter</title>
    </head>

    <body>
      <div id="topsection">
        <header>
          <h1>Le blog trotter</h1>
          <p>Je parcours la planète... et vous la fais découvrir !</p>
        </header>

        <ul id="menu">
          <li><a href="#">Accueil</a></li>
          <li><a href="#">Archives</a></li>
          <li><a href="#">Contact</a></li>
        </ul>
      </div>

      <h1>La Chine</h1>
      <p>a République populaire de Chine, couramment appelée Chine et parfois Chine </p>
      <p>populaire, est un pays d'Asie de l'Est.</p>

      <h1>L'Espagne</h1>
      <p>L'Espagne, en forme longue, le Royaume d'Espagne, en castillan España et Reino de España,</p>
      <p>est un pays d'Europe du Sud et, selon les définitions, de l'Ouest, qui occupe la plus </p>
      <p>grande partie de la péninsule Ibérique.</p>

      <footer>
        <p>Copyright Le Blog Trotter</p>
      </footer>

    </body>
</html>
```

À changer en Style Leblog.css

Modifier l'id en "Conteneur"

Dans la suite du travail des éléments devront être modifiés, ajoutés ou supprimés.

Créer aussi un fichier CSS nommé : Style Leblog pour l'instant vide.

C'est à toi maintenant de travailler et de trouver la solution des 4 exercices présentés à la page précédente.

Chapitre 5 : Solutions

- Une balise sémantique "nav" manque. Pouvez-vous l'ajouter au bon endroit ?
  Dans le fichier HTML

```
        <nav>
        <ul id="menu">
          <li><a href="#">Accueil</a></li>
          <li><a href="#">Archives</a></li>
          <li><a href="#">Contact</a></li>
        </ul>
        </nav>
      </div>
```

- Retirer les puces de la liste à puces (à vous de trouver comment faire !)
  Dans le fichier CSS

```
1  ul
2  {
3      list-style-type: none;
4  }
```

- Placer le header et le menu côte à côte. Créer une bordure autour de ces 2 éléments et ajouter une couleur de fond différente à chacun de ceux-ci.

Fichier HTML :

```
<div id="conteneur">
    <div class="element">
        <header>
            <h1>Le blog trotter</h1>
            <p>Je parcours la planète... et vous la fais découvrir !</p>
        </header>
    </div>

    <div class="element">
        <nav>
            <ul id="menu">
                <li><a href="#">Accueil</a></li>
                <li><a href="#">Archives</a></li>
                <li><a href="#">Contact</a></li>
            </ul>
        </nav>
    </div>
</div>
```

Fichier CSS :

```css
/* Feuille de style */

ul
{
    list-style-type: none;
}

#conteneur
{
    Display: flex;
    flex-direction: row;
    border: 2px solid #444;
}

.element
{
    width: 400px;
    height: 100px;
}

.element:nth-child(1)
{
    background-color: orange;
}

.element:nth-child(2)
{
    background-color: yellow;
}
```

- Le plus gros point à réaliser (en CSS):
  1. Afficher les paragraphes en justifié
  2. Définir une largeur et une hauteur bien spécifique aux paragraphes (550px, 120px)
  3. Centrer les paragraphes dans leur bloc
  4. Associer une couleur de fond identique pour les paragraphes
  5. Souligner les titres H1 des paragraphes en rouge (en CSS)
  6. Créer des espacements </br> afin d'éclaircir la page
  7. Choisir une police Comic Sans MS

Fichier HTML :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="Style Leblog.css" />
        <title>Le blog trotter</title>
    </head>

    <body>
    <div id="conteneur">
        <div class="element">
            <header>
                <h1>Le blog trotter</h1></br></br></br>
                Je parcours la planète... et vous la fais découvrir !
            </header>
        </div>
        <div class="element">
            <nav>
                <ul id="menu">
                    <li><a href="#">Accueil</a></li>
                    <li><a href="#">Archives</a></li>
                    <li><a href="#">Contact</a></li>
                </ul>
            </nav>
        </div>
     </div>
        </br></br></br>
            <h1>La Chine</h1>
            <p>La République populaire de Chine, couramment appelée Chine et parfois Chine
            populaire, est un pays d'Asie de l'Est.</p>

            <h1>L'Espagne</h1>
            <p>L'Espagne, en forme longue, le Royaume d'Espagne, en castillan España et Reino de España,
            est un pays d'Europe du Sud et, selon les définitions, de l'Ouest, qui occupe la plus
            grande partie de la péninsule Ibérique.</p>
    </br></br></br>
            <footer center>
                Copyright Le Blog Trotter
            </footer>
    </body>
</html>
```

Fichier CSS :

```
31  h1
32  {
33      font-size: 30px;
34      font-family: "Comic sans MS";
35      text-decoration: underline red;
36
37  }
38
39  p
40  {
41      text-align: center;
42      background-color: grey;
43      width: 550px;
44      height : 120px;
45      font-size: 15px;
46      font-family: "Comic sans MS";
47  }
```

Fonctionnalités Évoluées

Chapitre 6 : Les tableaux

Indispensables pour organiser les informations, les tableaux sont un peu délicats à construire en HTML. Il va en effet falloir imbriquer de nouvelles balises HTML dans un ordre précis.

Nous allons commencer par construire des tableaux basiques, puis nous les complexifierons au fur et à mesure : fusion de cellules, division en multiples sections, etc. Nous découvrirons aussi les propriétés CSS liées aux tableaux, qui nous permettront de personnaliser leur apparence.

Un tableau simple

La première balise à connaître est `<table> </table>`. C'est cette balise qui permet d'indiquer le début et la fin d'un tableau.

Cette balise est de type bloc, il faut donc la placer en dehors d'un paragraphe.

Voici un exemple que tu peux insérer dans un nouvel exercice HTML que tu peux intituler `Fonctionnalités évoluées`. N'oublie pas de créer aussi un fichier CSS de même nom qui sera associé au fichier HTML.

Fichier HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Les Fonctionnalités évoluées</title>
        <link rel="stylesheet" href="Fonctionnalités évoluées.css" />
    </head>

<body>
        <header>
            <h1>Kimps Bart</h1>
            <h2>Fonctionnalités évoluées</h2>
        </header>
        <p>
        Vous allez retrouver ici des exercices et exemples sur des fonctionnalisés évoluées en CSS
        </p>
        <nav>
            <ul>
                <li><a href="#"></a></li>
                <li><a href="#"></a></li>
                <li><a href="#"></a></li>
            </ul>
        </nav>

        <section>
            <aside>
                <h1>Auteur : Kimps Bart</h1>
                <p></p>
            </aside>

            <article>
                <h1>Matière à découvrir à travers des exercices</h1>
            </article>
        </section>
        <footer>
            <p>Copyright Bart - Tous droits réservés<br />
            <a href="#">Me contacter !</a></p>
        </footer>
    </body>
</html>
```

Le fichier CSS est vide.

Tu vas créer un tableau avec des élèves de mon cours en informatique + lien à partir de la section de ta page (menu) vers une ancre.

Ajoute donc maintenant ces lignes dans ton fichier HTML :

```
<nav>
    <ul>
        <li><a href="#tableau">Les Tableaux</a></li>
        <li><a href="#"></a></li>
        <li><a href="#"></a></li>
    </ul>
</nav>

<section>
    <table>
        <a id="tableau">Tableaux avec les élèves du cours informatique</a>
        <tr>
            <td>Carmen</td>
            <td>33 ans</td>
            <td>Espagne</td>
        </tr>
        <tr>
            <td>Michelle</td>
            <td>26 ans</td>
            <td>États-Unis</td>
    </table>

    <aside>
        <h1>Auteur : Kimps Bart</h1>
        <p></p>
    </aside>
```

Regarde le résultat à chaque étape.

Créer des bordures

Ajoutons des bordures à ton tableau. Ceci se réalise en CSS.

```
td /* Toutes les cellules des tableaux... */
{
    border: 1px solid black; /* auront une bordure de 1px */
}
```

Ensuite on colle les bordures.

```
table
{
    border-collapse: collapse; /* Les bordures du tableau seront collées (plus joli) */
}
```

La ligne d'en-tête

Ajoutons la ligne d'en-tête du tableau. Dans l'exemple ci-dessous, les en-têtes sont « Nom », « Âge » et « Pays ».
La ligne d'en-tête se crée avec un `<tr>` comme on l'a fait jusqu'ici, mais les cellules qu'elle contient sont, cette fois, encadrées par des balises `<th>` et non pas `<td>` !

```
<table>
    <a id="tableau">Tableaux a
    <tr>
        <th>Nom</th>
        <th>Âge</th>
        <th>Pays</th>
    </tr>
    <tr>
```

Appliquons une bordure aussi autour de ces cellules.

```
td, th /* Toutes les cellules des tableaux... */
{
    border: 1px solid black; /* auront une bordure de 1px */
}
```

Titre du tableau

Normalement, tout tableau doit avoir un titre. Le titre permet de renseigner rapidement le visiteur sur le contenu du tableau.
Notre exemple est constitué d'une liste de personnes… oui mais alors ? Qu'est-ce que cela représente ?

Sans titre de tableau, on est un peu perdu.

Heureusement, il y a `<caption>`!
Cette balise se place tout au début du tableau, juste avant l'en-tête. C'est elle qui contient le titre du tableau :

```
<section>
    <table>
        <a id="tableau"><caption>Tableaux avec les élèves du cours informatique</caption></a>
        <tr>
            <th>Nom</th>
            <th>Âge</th>
            <th>Pays</th>
        </tr>
```

Un tableau structuré

Nous avons appris à construire un petit tableau simple.

Nous allons découvrir deux techniques particulières pour construire un tableau plus complexe.

- Pour les gros tableaux, il est possible de les **diviser** en trois parties :
    - En-tête ;
    - Corps du tableau ;
    - Pied de tableau.
- Pour certains tableaux, il se peut que vous ayez besoin de fusionner des cellules entre elles.

Diviser un gros tableau

Si ton tableau est assez gros, tu auras tout intérêt à le découper en plusieurs parties. Pour cela, il existe des balises HTML qui permettent de définir les trois « zones » du tableau :

- **l'en-tête (en haut)** : il se définit avec les balises `<thead></thead>`;
- l**e corps (au centre)** : il se définit avec les balises `<tbody></tbody>`;
- **le pied du tableau (en bas)** : il se définit avec les balises `<tfoot></tfoot>`.

Que mettre dans le pied de tableau ? Généralement, si c'est un long tableau, vous y recopiez les cellules d'en-tête. Cela permet de voir, même en bas du tableau, à quoi se rapporte chacune des colonnes.

Tu trouveras à la page suivante le tableau que tu as créé précédemment et diviser. Des nouveaux élèves se sont ajoutés aux cours.

```
<a id="tableau"><caption>Tableaux avec les élèves du cours informatique</caption></a>
<thead> <!-- En-tête du tableau -->
<tr>
    <th>Nom</th>
    <th>Âge</th>
    <th>Pays</th>
</tr>
</thead>

<tfoot> <!-- Pied du tableau -->
<tr>
    <th>Nom</th>
    <th>Âge</th>
    <th>Pays</th>
</tr>
</tfoot>

<tbody> <!-- Corps du tableau -->
<tr>
    <td>Carmen</td>
    <td>33 ans</td>
    <td>Espagne</td>
</tr>
<tr>
    <td>Michelle</td>
    <td>26 ans</td>
    <td>États-Unis</td>
</tr>
<tr>
    <td>François</td>
    <td>43 ans</td>
    <td>France</td>
</tr>
<tr>
    <td>Martine</td>
    <td>34 ans</td>
    <td>France</td>
</tr>
<tr>
    <td>Jonathan</td>
    <td>13 ans</td>
    <td>Australie</td>
</tr>
<tr>
    <td>Xu</td>
    <td>19 ans</td>
    <td>Chine</td>
</tr>
```

Fusionner

Dans certains tableaux complexes, tu auras besoin de « fusionner » des cellules entre elles.
Un exemple de fusion ? Regardez le tableau à la figure suivante, qui dresse une liste de films et indique à qui ils s'adressent.

Modification de l'ancre pour le nouveau tableau et de l'ancienne ligne.

```
<nav>
    <ul>
        <li><a href="#tableau">Le tableau de mes élèves</a></li>
        <li><a href="#tableauF">Le tableau de mes films</a></li>
        <li><a href="#"></a></li>
    </ul>
</nav>
```

Ajout d'un nouveau tableau dans le fichier HTML en dessous du précédent.

```
            <td>Xu</td>
            <td>19 ans</td>
            <td>Chine</td>
        </tr>
</table>
</br>
<table>
<a id="tableauF"><caption>Le tableau de mes films</caption></a>
        <tr>
            <th>Titre du film</th>
            <th>Pour enfants ?</th>
            <th>Pour adolescents ?</th>
        </tr>
        <tr>
            <td>Massacre à la tronçonneuse</td>
            <td >Non, trop violent</td>
            <td>Oui</td>
        </tr>
        <tr>
            <td>Les bisounours font du ski</td>
            <td>Oui, adapté</td>
            <td>Pas assez violent...</td>
        </tr>
        <tr>
        <td>Lucky Luke, seul contre tous</td>
        <td>Pour toute la famille !</td>
        </tr>
</table>
```

Un élément manque pour Lucky Luke, une fusion de la dernière ligne sera réalisée à l'aide de <td colspan="2">Pour toute la famille !</td>.

- Colspan: fusion horizontale ;
- Rowspan: fusion verticale.

```
        <tr>
        <td>Lucky Luke, seul contre tous</td>
        <td colspan="2">Pour toute la famille !</td>
        </tr>
</table>
```

Exercice :

Crée maintenant un nouveau tableau qui sera inversé. Les titres seront à gauche et tu ne vas donc plus utiliser la fusion horizontale mais verticale.

Ajoute une ligne dans le menu de navigation et modifie les titres de ces menus pour qu'ils soient plus explicites.

Solution à la page suivante.

Kimps Bart

Fonctionnalités évoluées

Vous allez retrouver ici des exercices et exemples sur des fonctionnalisés évoluées en CSS

- Le tableau de mes élèves
- Le tableau de mes films en horizontale
- Le tableau de mes films en verticale

Tableaux avec les élèves du cours informatique

Nom	Âge	Pays
Carmen	33 ans	Espagne
Michelle	26 ans	États-Unis
François	43 ans	France
Martine	34 ans	France
Jonathan	13 ans	Australie
Xu	19 ans	Chine
Nom	**Âge**	**Pays**

Le tableau de mes films en horizontale

Titre du film	Pour enfants ?	Pour adolescents ?
Massacre à la tronçonneuse	Non, trop violent	Oui
Les bisounours font du ski	Oui, adapté	Pas assez violent...
Lucky Luke, seul contre tous	Pour toute la famille !	

Le tableau de mes films en verticale

Titre du film	Massacre à la tronçonneuse	Les bisounours font du ski	Lucky Luke, seul contre tous
Pour enfants ?	Non, trop violent	Oui, adapté	Pour toute la famille !
Pour adolescents ?	Oui	Pas assez violent...	

Auteur : Kimps Bart

Matière à découvrir à travers des exercices

Nouveau menu :

```
<nav>
    <ul>
        <li><a href="#tableau">Le tableau de mes élèves</a></li>
        <li><a href="#tableauF">Le tableau de mes films en horizontale</a></li>
        <li><a href="#tableauH">Le tableau de mes films en verticale</a></li>
    </ul>
</nav>
```

Nouveau tableau :

```
<table>
<a id="tableauH"><caption>Le tableau de mes films en verticale</caption></a>
    <tr>
        <th>Titre du film</th>
        <td>Massacre à la tronçonneuse</td>
        <td>Les bisounours font du ski</td>
        <td>Lucky Luke, seul contre tous</td>
    </tr>
    <tr>
        <th>Pour enfants ?</th>
        <td>Non, trop violent</td>
        <td>Oui, adapté</td>
        <td rowspan="2">Pour toute la famille !</td>
    </tr>
    <tr>
        <th>Pour adolescents ?</th>
        <td>Oui</td>
        <td>Pas assez violent...</td>
    </tr>
</table>
```

En résumé

- Un tableau s'insère avec la balise `<table>` et se définit ligne par ligne avec `<tr>`.
- Chaque ligne comporte des cellules `<td>` (cellules normales) ou `<th>` (cellules d'en-tête).
- Le titre du tableau se définit avec `<caption>`.
- On peut ajouter une bordure aux cellules du tableau avec border. Pour fusionner les bordures, on utilise la propriété CSS `border-collapse`.
- Un tableau peut être divisé en trois sections :`<thead>` (en-tête), `<tbody>` (corps) et `<tfoot>` (bas du tableau). L'utilisation de ces balises n'est pas obligatoire.
- On peut fusionner des cellules horizontalement avec l'attribut `colspan` ou verticalement ave `crowspan`. Il faut indiquer combien de cellules doivent être fusionnées.

Chapitre 7 : Les formulaires

Ce chapitre a déjà été travaillé dans le cours HTML donc le but de celui-ci est devoir d'une manière rapide ce qui a été vu et d'approfondir ce qui n'a pas encore été découvert.

Créer un formulaire

Voici un exemple que tu peux insérer dans un nouvel exercice HTML que tu peux intituler Les formulaires. N'oublie pas de créer aussi un fichier CSS de même nom qui sera associé au fichier HTML.

Tu peux y insérer ces instructions avec la méthode de traitement des données :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Les Formulaires</title>
        <link rel="stylesheet" href="Les formulaires.css" />
    </head>

<body>
        <header>
            <h1>Kimps Bart</h1>
            <h2>Les formulaires</h2>
        </header>
        <p>
        Vous allez retrouver ici des exercices et exemples sur les formulaires
        </p>
        <nav>
            <ul>
                <li><a href="#"></a></li>
                <li><a href="#"></a></li>
                <li><a href="#"></a></li>
            </ul>
        </nav>

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

            <aside>
                <h1>Auteur : Kimps Bart</h1>
                <p></p>
            </aside>

            <article>
                <h1>Matière à découvrir à travers des exercices</h1>
            </article>
        </section>
        <footer>
            <p>Copyright Bart - Tous droits réservés<br />
            <a href="#">Me contacter !</a></p>
        </footer>
    </body>
</html>
```

Zone de texte monoligne

Pour créer une zone de texte à une ligne et lui donner un nom :

```
<form method="post" action="traitement.php">
    <p><input type="text" name="pseudo" /></p>
</form>
```

Les libellés

Cette zone de texte est bien jolie mais si votre visiteur tombe dessus, il ne sait pas ce qu'il doit écrire. C'est justement le rôle de la balise :

```
<form method="post" action="traitement.php">
    <p><label>Votre pseudo</label> : <input type="text" name="pseudo" /></p>
</form>
```

Mais cela ne suffit pas. Il faut lier le label à la zone de texte.
Pour ce faire, on doit donner un nom à la zone de texte, non pas avec l'attribut `name` mais avec l'attribut `id` (que l'on peut utiliser sur toutes les balises).

Pour ne pas causer de problème il faut donner le même nom aux deux attributs (name et id).

Pour lier le label au champ, il faut lui donner un attribut `for` qui a la même valeur que l'`id` du champ…

Voici un complément à insérer dans la ligne de code :

```
<form method="post" action="traitement.php">
    <p><label for="pseudo">Votre pseudo</label> : <input type="text" name="pseudo" id="pseudo" /></p>
</form>
```

Quelques attributs supplémentaires

On peut ajouter un certain nombre d'autres attributs à la balise `<input />` pour personnaliser son fonctionnement :

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

Voici du code afin de compléter ton travail :

```
        <section>
        <form method="post" action="traitement.php">
            <p><label for="pseudo">Votre pseudo</label> : <input type="text" name="pseudo"
            id="pseudo" placeholder="Ex : Pseudo" size="30" maxlength="10"  /></p>
        </form>

            <aside>
```

Zone de mot de passe

Je complète mon formulaire. Il demande maintenant au visiteur son pseudo et son mot de passe :

```
<form method="post" action="traitement.php">
        <p><label for="pseudo">Votre pseudo</label> : <input type="text" name="pseudo"
        id="pseudo" placeholder="Ex : Pseudo" size="30" maxlength="10"  /></p>

        <br />
        <label for="pass">Votre mot de passe :</label>
        <input type="password" name="pass" id="pass" />
</form>
```

Zone de texte multiligne

```
        <br />
        <label for="ameliorer">Comment pensez-vous que je pourrais améliorer mon site ?</label><br /
        <textarea name="ameliorer" id="ameliorer"></textarea>
</form>
```

Ajout de ce texte dans la zone textarea

```
        <label for="ameliorer">Comment pensez-vous que je pourrais améliorer mon site ?</label><br /
        <textarea name="ameliorer" id="ameliorer" rows="10" cols="50">
        Améliorer ton site ?!
        Mais enfin ! Il est tellement génialissime
        qu'il n'est pas nécessaire de l'améliorer !
        </textarea>
</form>
```

Les zones de saisie enrichies

En voici quelques une :

```
        </textarea>
        <br />  <br />
        Zones enrichies
        <br />
        E-Mail : <input type="email" /> <br />
        URL : <input type="url" />  <br />
        Numéro de téléphone : <input type="tel" />
        Nombre : <input type="number" />     <br />
        Curseur : <input type="range" />     <br />
        Couleur : <input type="color" />     <br />
        Date : <input type="date" />     <br />
        Recherche : <input type="search" /> <br />
</form>
```

Les éléments d'options

HTML vous offre une ribambelle d'éléments d'options à utiliser dans votre formulaire. Ce sont des éléments qui demandent au visiteur de faire un choix parmi une liste de possibilités. Nous allons passer en revue :

1. les cases à cocher ;
2. les zones d'options ;
3. les listes déroulantes.

Les cases à cocher

```
        <p>
            Cochez les aliments que vous aimez manger :<br />
            <input type="checkbox" name="frites" id="frites" />
                <label for="frites">Frites</label><br />
            <input type="checkbox" name="steak" id="steak" />
                <label for="steak">Steak haché</label><br />
            <input type="checkbox" name="epinards" id="epinards" />
                <label for="epinards">Epinards</label><br />
            <input type="checkbox" name="huitres" id="huitres" />
                <label for="huitres">Huitres</label>
        </p>
</form>
```

Tu peux faire en sorte qu'une case soit cochée par défaut avec l'attribut checked.

Les zones d'options

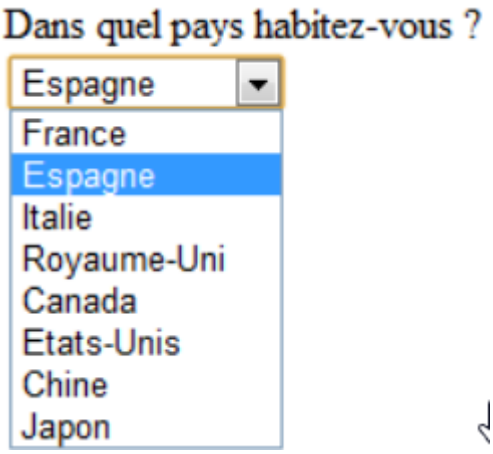
```
        <p>
            Veuillez indiquer la tranche d'âge dans laquelle vous vous situez :<br />
            <input type="radio" name="age" value="moins15" id="moins15" />
                <label for="moins15">Moins de 15 ans</label><br />
            <input type="radio" name="age" value="medium15-25" id="medium15-25" />
                <label for="medium15-25">15-25 ans</label><br />
            <input type="radio" name="age" value="medium25-40" id="medium25-40" />
                <label for="medium25-40">25-40 ans</label><br />
            <input type="radio" name="age" value="plus40" id="plus40" />
                <label for="plus40">Encore plus vieux que ça ?!</label>
        </p>
</form>
```

Les boutons font tous partis d'un même groupe. Chaque zone d'options différente doit posséder un nom unique.

Les listes déroulantes

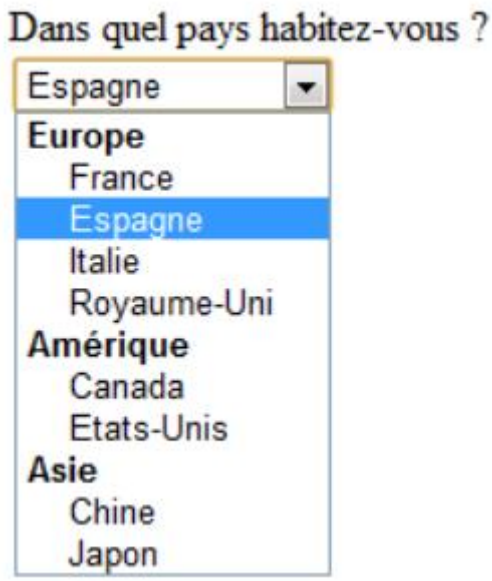
```
        <p>
            <select name="pays" id="pays">
                <option value="france">France</option>
                 <option value="espagne">Espagne</option>
                <option value="italie">Italie</option>
                <option value="royaume-uni">Royaume-Uni</option>
                <option value="canada">Canada</option>
                <option value="etats-unis">États-Unis</option>
                <option value="chine">Chine</option>
                <option value="japon">Japon</option>
            </select>
        </p>
</form>
```

Si vous voulez qu'une option soit sélectionnée par défaut, utilisez cette fois l'attribut selected.



On peut aussi grouper les options :

```
        <p>
        <label for="pays">Dans quel pays habitez-vous ?</label><br />
        <select name="pays" id="pays">
            <optgroup label="Europe">
                <option value="france">France</option>
                <option value="espagne">Espagne</option>
                <option value="italie">Italie</option>
                <option value="royaume-uni">Royaume-Uni</option>
            </optgroup>
            <optgroup label="Amérique">
                <option value="canada">Canada</option>
                <option value="etats-unis">Etats-Unis</option>
            </optgroup>
                <optgroup label="Asie">
                <option value="chine">Chine</option>
                <option value="japon">Japon</option>
            </optgroup>
        </select>
    </p>
</form>
```



Finaliser et envoyer le formulaire

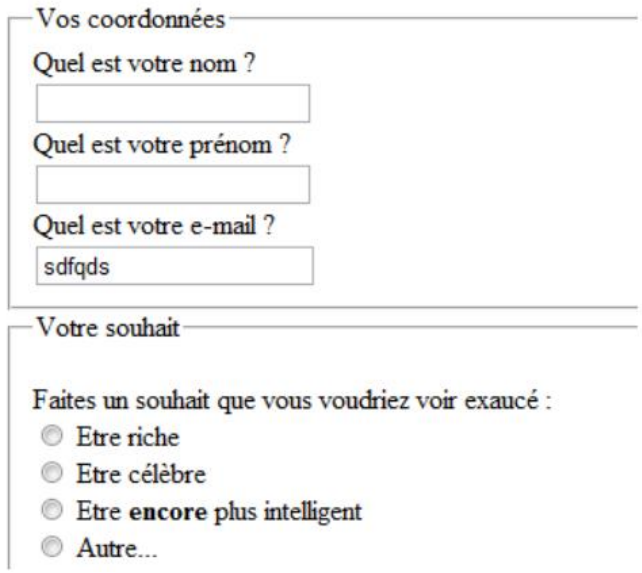
Nous y sommes presque. Il ne nous reste plus qu'à agrémenter notre formulaire de quelques dernières fonctionnalités (comme la validation), puis nous pourrons ajouter le bouton d'envoi du formulaire.

Regrouper les champs

Si ton formulaire grossit et comporte beaucoup de champs, il peut être utile de les regrouper au sein de plusieurs balises `<fieldset>`. Chaque `<fieldset>` peut contenir une légende avec la balise `<legend>`. Regarde cet exemple que tu insères dans ton travail :

```
        <fieldset>
            <legend>Votre souhait</legend> <!-- Titre du fieldset -->
            <p>
                Faites un souhait que vous voudriez voir exaucé :
                <input type="radio" name="souhait" value="riche" id="riche" />
                    <label for="riche">Etre riche</label>
                <input type="radio" name="souhait" value="celebre" id="celebre" />
                    <label for="celebre">Etre célèbre</label>
                <input type="radio" name="souhait" value="intelligent" id="intelligent" />
                    <label for="intelligent">Etre <strong>encore</strong> plus intelligent</label>
                <input type="radio" name="souhait" value="autre" id="autre" />
                    <label for="autre">Autre...</label>
            </p>
            <p>
                <label for="precisions">Si "Autre", veuillez préciser :</label>
                <textarea name="precisions" id="precisions" cols="40" rows="4"></textarea>
            </p>
        </fieldset>
</form>
```

Tu vas obtenir ce résultat :



Place le curseur dans l'un des champs à l'aide l'attribut `autofocus`.

À l'aide de l'attribut required, un champ sera obligatoire.

Et pour qu'un champ obligatoire soit d'une autre couleur de fond, tu peux ajouter dans ton fichier CSS :

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

Bouton d'envoi

<input type="submit" value="Envoyer" />

Le problème, c'est que vous ne pouvez pas créer cette page seulement en HTML. Il est nécessaire d'apprendre un nouveau langage, comme le PHP, pour pouvoir « récupérer » les informations saisies et décider quoi en faire.

Voir Cahiers d'exercice : Cours en PHP.

En résumé

- Un formulaire est une zone interactive de la page, dans laquelle les visiteurs peuvent saisir des informations.
- On délimite un formulaire avec la balise <form> à laquelle il faut ajouter deux attributs : method (mode d'envoi des données) et action (page vers laquelle le visiteur sera redirigé après envoi du formulaire et qui traitera les informations).
- Une grande partie des éléments du formulaire peut s'insérer avec la balise <input />. La valeur de son attribut type permet d'indiquer quel type de champ doit être inséré :
    - text : zone de texte ;
    - password : zone de texte pour mot de passe ;
    - tel : numéro de téléphone ;
    - checkbox : case à cocher ;
    - etc.
- La balise <label> permet d'écrire un libellé. On l'associe à un champ de formulaire avec l'attribut for, qui doit avoir la même valeur que l'id du champ de formulaire.

- On peut rendre un champ obligatoire avec l'attribut required, faire en sorte qu'il soit sélectionné par défaut avec autofocus, donner une indication dans le champ avec placeholder…
- Pour récupérer ce que les visiteurs ont saisi, le langage HTML ne suffit pas. Il faut utiliser un langage « serveur » comme PHP.

Chapitre 8 : La vidéo et l'audio

Depuis l'arrivée de Youtube et Dailymotion, il est devenu courant aujourd'hui de regarder des vidéos sur des sites web.

Cependant, aucune balise HTML ne permettait jusqu'ici de gérer la vidéo. Il fallait à la place utiliser un plugin, comme Flash. Celui-ci reste encore en partie utilisé pour regarder des vidéos sur Youtube, Dailymotion, Vimeo et ailleurs. Mais utiliser un plugin a de nombreux défauts : on dépend de ceux qui gèrent le plugin (en l'occurrence, l'entreprise Adobe, qui possède Flash), on ne peut pas toujours contrôler son fonctionnement, il y a parfois des failles de sécurité… Au final, c'est assez lourd.

C'est pour cela que deux nouvelles balises standard ont été créées en HTML5 : <video> et <audio>!

Les formats audio et vidéo

Les formats audio

Pour diffuser de la musique ou n'importe quel son, il existe de nombreux formats. La plupart d'entre eux sont compressés (comme le sont les images JPEG, PNG et GIF) ce qui permet de réduire leur poids :

- **MP3** : C'est l'un des plus vieux, mais aussi l'un des plus compatibles (tous les appareils savent lire des MP3), ce qui fait qu'il est toujours très utilisé aujourd'hui.
- **AAC** : utilisé majoritairement par Apple sur iTunes, c'est un format de bonne qualité. Les iPod, iPhone et autres iPad savent les lire sans problème.
- **OGG** : le format Ogg Vorbis est très répandu dans le monde du logiciel libre, notamment sous Linux. Ce format a l'avantage d'être libre, c'est-à-dire qu'il n'est protégé par aucun brevet.
- **WAV** (**format non compressé**) : évitez autant que possible de l'utiliser car le fichier est très volumineux avec ce format.

La compatibilité dépend des navigateurs, mais elle évolue dans le bon sens au fil du temps. Pense à consulter Caniuse.com pour connaître la compatibilité actuelle du MP3, AAC, OGG, WAV…

Les formats vidéo

Le stockage de la vidéo est autrement plus complexe. On a besoin de trois éléments :

- **Un format conteneur** : c'est un peu comme une boîte qui va servir à contenir les deux éléments ci-dessous. On reconnaît en général le type de conteneur à l'extension du fichier : AVI, MP4, MKV…
- **Un codec audio** : c'est le format du son de la vidéo, généralement compressé. Nous venons de les voir, on utilise les mêmes : MP3, AAC, OGG…
- **Un codec vidéo** : c'est le format qui va compresser les images. C'est là que les choses se corsent, car ces formats sont complexes et on ne peut pas toujours les utiliser gratuitement. Les principaux à connaître pour le Web sont :
    - **H.264** : l'un des plus puissants et des plus utilisés aujourd'hui… mais il n'est pas 100% gratuit. En fait, on peut l'utiliser gratuitement dans certains cas (comme la diffusion de vidéos sur un site web personnel), mais il y a un flou juridique qui fait qu'il est risqué de l'utiliser à tout va.
    - **Ogg Theora** : un codec gratuit et libre de droits, mais moins puissant que H.264. Il est bien reconnu sous Linux mais, sous Windows, il faut installer des programmes pour pouvoir le lire.
    - **WebM** : un autre codec gratuit et libre de droits, plus récent. Proposé par Google, c'est le concurrent le plus sérieux de H.264 à l'heure actuelle.

Là encore, surveille bien la compatibilité sur Caniuse.com : H.264, Ogg Theora, WebM... Le format H.264 semble sortir du lot. Il est quand même conseillé si possible de proposer chaque vidéo dans plusieurs formats pour qu'elle soit lisible sur un maximum de navigateurs.

Pour convertir une vidéo dans ces différents formats, je te conseille l'excellent logiciel gratuit Miro Video Converter (pour Mac OS X – pour Windows).

Il vous suffit de glisser-déposer votre vidéo dans la fenêtre du programme et de sélectionner le format de sortie souhaité. Cela vous permettra de créer plusieurs versions de ta vidéo !

Insertion d'un élément audio

Voici la balise permettant de jouer du son sur ton PC :

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

Mais tu dois compléter la balise des attributs suivants :

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

Crée un nouveau fichier HTML comme celui-ci-dessous afin de tester cette balise :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Audio vidéo</title>
        <link rel="stylesheet" href="" />
    </head>

<body>
        <header>
            <h1>Kimps Bart</h1>
            <h2>Audio vidéo</h2>
        </header>
        <p>
        Vous allez retrouver ici des exercices et exemples sur les balises Audio vidéo
        </p>
        <nav>
            <ul>
                <li><a href="#Audio">Exercices sur l'audio</a></li>
                <li><a href="#"></a></li>
                <li><a href="#"></a></li>
            </ul>
        </nav>

        <section>
            <a id="Audio"><caption>Exercices sur l'audio</caption></a>
            <audio src="hype_home.mp3" controls></audio>

            <aside>
                <h1>Auteur : Kimps Bart</h1>
                <p></p>
            </aside>

            <article>
                <h1>Matière à découvrir à travers des exercices</h1>
            </article>
        </section>
        <footer>
            <p>Copyright Bart - Tous droits réservés<br />
            <a href="#">Me contacter !</a></p>
        </footer>
    </body>
</html>
```

Trouve un fichier audio et adapte le code en fonction de ton choix. L'audio doit-être dans un dossier Audio pour l'exercice Teste la page.

Voici ce que tu obtiens en lançant ta page à partir de Google Chrome.

Microsoft Edge :

Si le navigateur ne gère pas le mp3 tu peux aussi proposer plusieurs versions du fichier audio en écrivant :

```
<section>
    <a id="Audio"><caption>Exercices sur l'audio</caption></a>
    <audio controls>
        <source src="hype_home.mp3">
        <source src="hype_home.ogg">
    </audio>
```

Insertion d'une vidéo

Voici la balise permettant de jouer une vidéo sur ton PC :

`<video src="sintel.webm"></video>`

Mais tu dois compléter la balise des attributs suivants :

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

Voici un exemple :

```
<section>
    <a id="Audio"><caption>Exercices sur l'audio</caption></a>
    <audio src="hype_home.mp3" controls></audio>
    <br/>
    <a id="Vidéo"><caption>Exercices sur la vidéo</caption></a>
    <video src="sintel.webm" controls poster="sintel.jpg" width="600"></video>

    <aside>
        <h1>Auteur : Kimps Bart</h1>
```

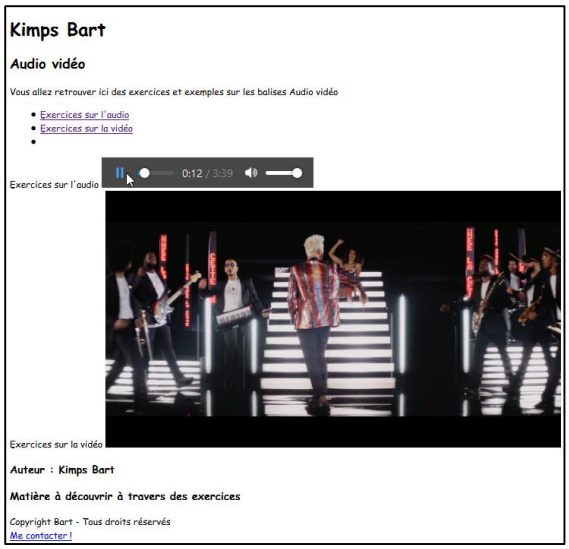
Trouve un fichier vidéo et adapte le code en fonction de ton choix. La vidéo doit-être dans un dossier Vidéo pour l'exercice. Teste la page.

Tu peux utiliser la balise `<source>` à l'intérieur de la balise `<video>` pour proposer différents formats. Le navigateur prendra celui qu'il reconnaît :

```
<section>
    <a id="Audio"><caption>Exercices sur l'audio</caption></a>
    <audio controls>
        <source src="hype_home.mp3">
        <source src="hype_home.ogg">
    </audio>
    <br/>
    <a id="Vidéo"><caption>Exercices sur la vidéo</caption></a>
    <video controls poster="sintel.jpg" width="600">
        <source src="sintel.mp4">
        <source src="sintel.webm">
        <source src="sintel.ogv">
    </video>

    <aside>
```

Source modifiée dans l'exercice précédent

Voici le travail terminé ouvert à l'aide de Firefox :



En résumé

- Insérer de la musique ou de la vidéo n'était pas possible autrefois en HTML. Il fallait recourir à un plugin comme Flash.
- Depuis HTML5, les balises <audio> et <video> ont été introduites et permettent de jouer de la musique et des vidéos sans plugin.
- Il existe plusieurs formats audio et vidéo. Il faut notamment connaître :
- pour l'audio : MP3 et Ogg Vorbis ;
- pour la vidéo : H.264, Ogg Theora et WebM.
- Aucun format n'est reconnu par l'ensemble des navigateurs : il faut proposer différentes versions de sa musique ou de sa vidéo pour satisfaire tous les navigateurs.
- Il faut ajouter l'attribut controls aux balises <audio> et <video> pour permettre au visiteur de lancer ou d'arrêter le média.
- Ces balises ne sont pas conçues pour empêcher le téléchargement de la musique et de la vidéo. Tu ne peux pas protéger ton média contre la copie.

Chapitre 9 : Le responsive design avec les Media Queries

Sais-tu quelle est la première préoccupation des webmasters qui mettent en place le design de leur site ? La résolution d'écran de leurs visiteurs. Eh oui : selon les écrans, il y a plus ou moins de place, plus ou moins de pixels de largeur.

Cette information est importante lorsque tu construis un design : comment ton site doit-il s'afficher en fonction des différentes résolutions d'écran ? Si tu as un écran large, tu risques d'oublier que certaines personnes naviguent avec des écrans plus petits. Et je ne te parle même pas des navigateurs des smartphones, qui sont encore moins larges.

C'est là que les media queries entrent en jeu. Ce sont des règles à appliquer pour changer le design d'un site en fonction des caractéristiques de l'écran ! Grâce à cette technique, nous pourrons créer un design qui s'adapte automatiquement à l'écran de chaque visiteur !

Mise en place des media queries

Les media queries font partie des nouveautés de CSS3. Il ne s'agit pas de nouvelles propriétés mais de *règles* que l'on peut appliquer dans certaines conditions. Concrètement, tu auras le pouvoir de dire « Si la résolution de l'écran du visiteur est inférieure à tant, alors applique les propriétés CSS suivantes ». Cela te permet de changer l'apparence du site dans certaines conditions : tu pourras augmenter la taille du texte, changer la couleur de fond, positionner différemment menu dans certaines résolutions, etc.
Contrairement à ce qu'on pourrait penser, les media queries ne concernent pas que les résolutions d'écran. Tu peux changer l'apparence de ton site en fonction d'autres critères comme le type d'écran (smartphone, télévision, projecteur…), le nombre de couleurs, l'orientation de l'écran (portrait ou paysage), etc. Les possibilités sont très nombreuses !

Appliquer une media query

Les media queries sont donc des règles qui indiquent quand on doit appliquer des propriétés CSS.

Il y a deux façons de les utiliser :

- en chargeant une feuille de style .css différente en fonction de la règle (ex : « Si la résolution est inférieure à 1280px de large, charge le fichier petite_resolution.css ») ;
- en écrivant la règle directement dans le fichier .css habituel (ex : « Si la résolution est inférieure à 1280px de large, charge les propriétés CSS ci-dessous »).

Chargement d'une feuille de style différente

Tu te souviens de la balise <link /> qui permet, dans notre code HTML, de charger un fichier .css ?

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

On peut lui ajouter un attribut media, dans lequel on va écrire la règle qui doit s'appliquer pour que le fichier soit chargé. On dit qu'on fait une « requête de media » (*media query* en anglais). Voici un exemple :

```
<link rel="stylesheet" media="screen and (max-width: 1280px)"
href="petite_resolution.css" />
```

Ceci a été ajouté à l'exercice précédent :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Audio vidéo et media queries</title>
        <link rel="stylesheet" href="" />
            <!--Pour tout le monde -->
        <link rel="stylesheet" media="screen and (max-width: 1280px)" href="petite_resolution.css" />
            <!-- Pour ceux qui ont une résolution inférieure à 1280px -->
    </head>
```

Chargement des règles directement dans la feuille de style

Une autre technique plus pratique, consiste à écrire ces règles dans le même fichier CSS que d'habitude.
Dans ce cas, on écrit la règle dans le fichier .css comme ceci :

```
@media screen and (max-width: 1280px)
{
    /* Rédigez vos propriétés CSS ici */
}
```

Modifie ton fichier HTML pour qu'une seule référence se réalise vers le fichier CSS.

```
<html>
    <head>
        <meta charset="utf-8" />
        <title>Audio vidéo et media queries</title>
        <link rel="stylesheet" href="media queries" />
    </head>
```

Les règles disponibles

Il existe de nombreuses règles permettant de construire des media queries. Je vous présente ici les principales :

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

NB : On peut rajouter le préfixe min- ou max- devant la plupart de ces règles. Ainsi, min-width signifie « Largeur minimale », max-height « Hauteur maximale », etc.
La différence entre width et device-width se perçoit surtout sur les navigateurs mobiles des smartphones, nous en reparlerons plus loin.

Les règles peuvent être combinées à l'aide des mots suivants :

- only : « uniquement » ;
- and : « et » ;
- not : « non ».

Voici des exemples :

```
/* Sur les écrans, quand la largeur de la fenêtre fait au maximum 1280px */
@media screen and (max-width: 1280px)

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

/* Sur les téléviseurs */
@media tv

/* Sur tous types d'écrans orientés verticalement */
@media all and (orientation: portrait)
```

Tester les media queries

Les media queries sont surtout utilisées pour adapter le design du site aux différentes largeurs d'écran.

Un test tout simple : changer la couleur et la taille du texte si la fenêtre fait plus ou moins de 1024 pixels de large. Pour ce test, je vais utiliser la seconde méthode qui consiste à écrire la règle directement dans le même fichier .css que d'habitude :

```
/* Paragraphes en bleu par défaut */

p
{
    color: blue;
}


/* Nouvelles règles si la fenêtre fait au plus 1024px de large */

@media screen and (max-width: 1024px)
{
    p
    {
        color: red;
        background-color: black;
        font-size: 1.2em;
    }
}
```

Si tu as envie de voir le fonctionnement de cette règle, introduit ces lignes dans ton fichier CSS Mon Site.css

Ouvre ton fichier Mon Site.html et modifie la grandeur de la fenêtre en appuyant sur la touche CTRL + la roulette de ta souris (ferme un zoom).

Que se passe-t-il ?

Mise en pratique des media queries sur le design

Télécharge l'exercice tp-final. Tu vas y travailler.

La page

Ouvre le fichier Style.css. Pour le moment, la largeur de la page est fixée à 900 px et le contenu est centré :

```
36  #bloc_page
37  {
38      width: 900px;
39      margin: auto;
40  }
```

À la suite de ces lignes, je te propose d'ajouter la règle media query suivante :

```
@media all and (max-width: 1024px)
{
    #bloc_page
    {
        width: auto;
    }
}
```

La règle signifie : « Pour tous les types d'écrans, si la largeur de la fenêtre ne dépasse pas 1024 px, alors exécuter les règles CSS suivantes ».

Les règles CSS en question sont très simples, il n'y en a en fait qu'une seule : on donne une largeur automatique à la page (plutôt qu'une largeur fixe de 900 px). La page prendra alors tout l'espace disponible dans la fenêtre. Cela évite l'apparition de barres de défilement horizontales sur les petites résolutions.

NB : auto est la valeur par défaut de la propriété width. Par défaut, les blocs ont une largeur automatique (ils prennent toute la place disponible). Cette valeur « écrase » celle que nous avions forcée à 900px quelques lignes plus haut : nous revenons donc au comportement par défaut du bloc.

Le menu de navigation

Nous voulons que le menu de navigation prenne moins de place sur les petites résolutions. Plutôt que de lui donner une dimension fixe, nous allons lui redonner sa dimension automatique flexible d'origine. Chaque élément du menu s'écrira en dessous du précédent : pour cela, nous demandons à ce que les éléments de la Flexbox soit organisés en colonne.

Enfin, le texte sera écrit plus petit et nous retirons la bordure en bas des liens lors du survol, car elle est moins adaptée à cette disposition.

```
@media all and (max-width: 1024px)
{
    #bloc_page
    {
        width: auto;
    }
    nav
    {
        width: auto;
        text-align: left;
    }

    nav ul
    {
        flex-direction: column;
    }

    nav li
    {
        padding-left: 4px;
    }

    nav a
    {
        font-size: 1.1em;
    }

    nav a:hover
    {
        border-bottom: 0;
    }
}
```

La bannière

Pour retirer la bannière, rien de plus simple : nous utilisons la propriété display à laquelle nous affectons la valeur none. Si la fenêtre est trop petite, nous préférons masquer complètement la bannière :

```
    nav a:hover
    {
        border-bottom: 0;
    }

    #banniere_image
    {
        display: none;
    }
}
```

Le bloc « À propos de l'auteur »

Plutôt que de placer ce bloc à droite de l'article, nous allons le faire passer en-dessous grâce à des Flexbox en colonne. Ce type de disposition « de haut en bas » est plus adapté aux petits écrans.

À l'intérieur du bloc, nous réajustons un peu la position des éléments : la photo de Zozor, notamment, sera placée en flottant à droite.

```
    #banniere_image
    {
        display: none;
    }

    section
    {
        flex-direction: column;
    }

    article, aside
    {
        width: auto;
        margin-bottom: 15px;
    }

    #fleche_bulle
    {
        display: none;
    }

    #photo_zozor img
    {
        width: 110px;
        float: right;
        margin-left: 15px;
    }

    aside p:last-child
    {
        text-align: center;
    }
}
```

NB : aside est un sélecteur avancé que nous n'avons pas utilisé jusqu'ici. aside p signifie « Tous les paragraphes à l'intérieur de la balise <aside> ». Avec :last-child, on cible uniquement le dernier paragraphe dans le bloc aside (celui qui contient les liens vers Facebook et Twitter), pour pouvoir centrer les images. Bien entendu, on aurait aussi pu affecter une class ou un id à ce paragraphe pour le cibler directement, mais je n'ai pas voulu modifier le code HTML.

Teste ton code.

En résumé

- Les media queries permettent de charger des styles CSS différents en fonction de certains paramètres.
- Les paramètres autorisés par les media queries sont nombreux : nombre de couleurs, résolution de l'écran, orientation… En pratique, on s'en sert surtout pour modifier l'apparence du site en fonction des différentes résolutions d'écran.
- On crée une media query avec la directive @media suivie du type d'écran et d'une ou plusieurs conditions (comme la largeur maximale d'écran). Le style CSS qui suit sera activé uniquement si les conditions sont remplies.
- Les navigateurs mobiles simulent une largeur d'écran : on appelle cela le viewport.
- On peut cibler les smartphones grâce à une règle basée sur le nombre réel de pixels affichés à l'écran : max-device-width.

Chapitre 10 : Aller plus loin

Alors que ce cours touche à sa fin, la tentation est grande de penser que l'on a tout vu. Tout vu ? Vous n'avez quand même pas cru cela ? Allons bon, il vous reste des centaines de choses à découvrir, que ce soit sur HTML, CSS, ou les technologies qui y sont liées (PHP, JavaScript…).

Ce chapitre a pour but de vous donner quelques directions pour compléter votre apprentissage. Alors ne soyez pas tristes, car vous n'avez pas fini de faire des découvertes !

Du site web à l'application web (JavaScript, AJAX…)

JavaScript est un langage qui existe depuis de nombreuses années maintenant et que l'on utilise fréquemment sur le Web en plus de HTML et CSS. C'est probablement l'un des premiers langages que tu voudras apprendre maintenant que tu as des connaissances en HTML et CSS.

À quoi JavaScript peut-il bien servir ? On ne peut pas tout faire avec HTML et CSS ?

On peut faire déjà beaucoup de choses en HTML et CSS mais, lorsqu'on veut rendre sa page plus interactive, un langage comme JavaScript devient indispensable.

Voici quelques exemples de ce à quoi peut servir JavaScript :

- On l'utilisera le plus souvent pour modifier des propriétés CSS sans avoir à recharger la page. Par exemple, vous pointez sur une image et le fond de votre site change de couleur (ce n'est pas possible à faire avec un :hover car cela concerne deux balises différentes, c'est bien là une limite du CSS).
- On peut l'utiliser aussi pour modifier le code source HTML sans avoir à recharger la page, *pendant* que le visiteur consulte la page.
- Il permet aussi d'afficher des boîtes de dialogue à l'écran du visiteur…
- … ou encore de modifier la taille de la fenêtre.

JavaScript est un langage qui se rapproche des langages de programmation tels que le C, C++, Python, Ruby… À l'inverse, HTML et CSS sont davantage des langages de description : ils décrivent comment la page doit apparaître mais ils ne donnent pas d'ordres directs à l'ordinateur (« fais ceci, fais cela… »), contrairement à JavaScript.

JavaScript n'a aucun rapport avec le langage Java. Seuls les noms se ressemblent.

JavaScript est régulièrement utilisé aujourd'hui pour faire de l'AJAX (Asynchronous JavaScript And XML). Cette technique permet de modifier une partie de la page web que le visiteur consulte en échangeant des données avec le serveur. Cela donne l'impression que les pages sont plus dynamiques et plus réactives. Le visiteur n'a plus besoin de recharger systématiquement toute la page.

Les navigateurs sont de plus en plus efficaces dans leur traitement de JavaScript, ce qui fait que les pages qui utilisent JavaScript sont de plus en plus réactives. On peut ainsi arriver aujourd'hui à créer des sites qui deviennent littéralement des applications web, l'équivalent de logiciels mais disponibles sous forme de sites web !
Un exemple célèbre : Google Docs, la suite bureautique de Google, disponible sur le Web.

Si tu as envie de découvrir du Javascript, tu peux retrouver des cahiers d'exercices à ce sujet.

Technologies liées à HTML5 (Canvas, SVG, Web Sockets…)

Le W3C ne travaille pas que sur les langages HTML et CSS. Ce sont certes les plus connus, mais le W3C cherche aussi à définir d'autres technologies qui viennent compléter HTML et CSS. Elles sont nombreuses et on les confond d'ailleurs souvent avec HTML5.

En fait, HTML5 est devenu un mot très utilisé qui fait référence à d'autres technologies que HTML. Quand quelqu'un vous parle de « HTML5 » aujourd'hui, il fait peut-être aussi référence à d'autres éléments qui sortent du cadre strict du HTML.

Voici une petite liste de ces nouvelles technologies introduites en parallèle de HTML5 (notez que certaines ne sont pas vraiment « nouvelles » mais elles reviennent sur le devant de la scène) :

- **Canvas** : permet de dessiner au sein de la page web, à l'intérieur de la balise HTML <canvas>. On peut dessiner des formes (triangles, cercles…) mais aussi ajouter des images, les manipuler, appliquer des filtres graphiques… Au final, cela nous permet de réaliser aujourd'hui de véritables jeux et des applications graphiques directement dans des pages web !
- **SVG** : permet de créer des dessins vectoriels au sein des pages web. À la différence de Canvas, ces dessins peuvent être agrandis à l'infini (c'est le principe du vectoriel). Le logiciel Inkscape est connu pour permettre de dessiner des SVG.
- **Drag & Drop** : permet de faire « glisser-déposer » des objets dans la page web, de la même façon qu'on peut faire glisser-déposer des fichiers sur son bureau. Gmail l'utilise pour permettre d'ajouter facilement des pièces jointes à un e-mail.
- **File API** : permet d'accéder aux fichiers stockés sur la machine du visiteur (avec son autorisation). On l'utilisera notamment en combinaison avec le Drag & Drop.
- **Géolocalisation** : pour localiser le visiteur et lui proposer des services liés au lieu où il se trouve (ex. : les horaires des salles de cinéma proches). La localisation n'est pas toujours très précise, mais cela peut permettre de repérer un visiteur à quelques kilomètres près (avec son accord).
- **Web Storage** : permet de stocker un grand nombre d'informations sur la machine du visiteur. C'est une alternative plus puissante aux traditionnels cookies. Les informations sont hiérarchisées, comme dans une base de données.
- **Appcache** : permet de demander au navigateur de mettre en cache certains fichiers, qu'il ne cherchera alors plus à télécharger systématiquement. Très utile pour créer des applications web qui peuvent fonctionner même en mode « hors ligne » (déconnecté).
- **Web Sockets** : permet des échanges plus rapides, en temps réel, entre le navigateur du visiteur et le serveur qui gère le site web (c'est une sorte d'AJAX amélioré). C'est un peu l'avenir des applications web, qui pourront devenir aussi réactives que les vrais programmes.

- **WebGL** : permet d'introduire de la 3D dans les pages web, en utilisant le standard de la 3D OpenGL (figure suivante). Les scènes 3D sont directement gérées par la carte graphique.

La plupart de ces technologies s'utilisent avec JavaScript. Il s'agit donc de nouvelles fonctionnalités que l'on peut utiliser en JavaScript.

Comme tu le vois, tu as de nouveaux mondes à découvrir ! Dès que tu connaisez suffisamment JavaScript, tu peux aller encore plus loin dans la gestion de ton site web… que tu peux même transformer en véritable application !

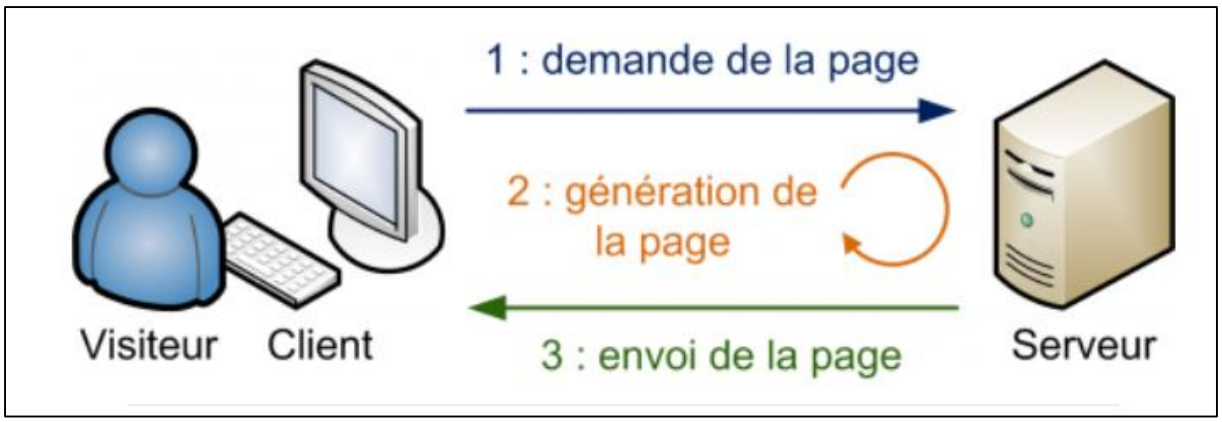
Les sites web dynamiques (PHP, JEE, ASP .NET…)

Les langages dont nous allons parler ici sont eux aussi des langages de programmation. Comme JavaScript ? Oui, mais avec une différence importante : JavaScript s'exécute sur la machine de tes visiteurs, tandis que les langages que nous allons voir s'exécutent sur le « serveur » qui contient ton site web.

Quelle différence cela fait-il que le programme tourne sur la machine du visiteur ou sur le serveur ?

Les différences sont importantes. Tout d'abord, en termes de puissance, un serveur sera bien souvent plus rapide que la machine de tes visiteurs, ce qui permet d'effectuer des calculs plus complexes. Tu as aussi davantage de contrôle côté serveur qu'en JavaScript… Mais le JavaScript reste irremplaçable car il y a certaines actions que tu ne peux faire que du côté « visiteur ».

Les langages serveur permettent de générer la page web lorsque le visiteur arrive sur ton site (figure suivante). Chaque visiteur peut donc obtenir une page web personnalisée suivant ses besoins !



Les langages ne servent donc pas aux mêmes choses, mais ils se complètent. Si tu combines HTML + CSS + JavaScript + PHP, par exemple, tu peux faire de l'AJAX (échanges de données entre la page et le serveur), tu peux effectuer des calculs, stocker des informations dans des bases de données… bref, faire de vrais sites web dynamiques !

Les langages « côté serveur » sont nombreux. Citons-en quelques-uns :

- **PHP** : l'un des plus connus. Facile à utiliser et puissant, il est utilisé notamment par Facebook… et OpenClassrooms. Un cahiers sur PHP peut être consulté si tu as envie d'apprendre plus.
- **Java EE** (Java) : très utilisé dans le monde professionnel, il s'agit d'une extension du langage Java qui permet de réaliser des sites web dynamiques, puissants et robustes. Au début, il est un peu plus complexe à prendre en main que PHP.
- **ASP .NET** (C#) : assez semblable à JEE, c'est le langage de Microsoft. On l'utilise en combinaison avec d'autres technologies Microsoft (Windows Server…). Il utilise le puissant framework .NET, véritable couteau suisse des développeurs, qui offre de nombreuses fonctionnalités.
- **Django** (Python) : une extension du langage Python qui permet de réaliser rapidement et facilement des sites web dynamiques. Il est connu pour générer des interfaces d'administration prêtes à l'emploi.
- **Ruby on Rails** (Ruby) : une extension du langage Ruby, assez similaire à Django, qui permet de réaliser des sites web dynamiques facilement et avec une grande souplesse.

Connaître l'un de ces langages est indispensable si tu veux traiter le résultat des formulaires HTML ! Souviens-toi de la balise <form> : je t'avais expliqué comment créer des formulaires, mais pas comment *récupérer* les informations saisies par tes visiteurs. Il te faut obligatoirement un langage serveur, comme PHP, pour récupérer et traiter ces données !

Au final, ces langages te permettent de réaliser des rêves les plus fous sur ton site web :

- forums ;
- newsletter ;
- compteur de visiteurs ;
- système de news automatisé ;

- gestion de membres ;
- jeux web (jeux de stratégie, élevage d'animaux virtuels…) ;
- etc.

Il est indispensable de connaître les langages HTML et CSS avant d'apprendre un langage serveur comme PHP !

Bonne découverte, si tu as envie !

Annexes

Chapitre 11 : Envoyer ton site Web

Une fois un site terminé, il ne doit pas rester rien que sur votre disque dur car personne d'autre ne va pouvoir en profiter.

Il faut donc l'envoyer sur le Web, mais… vous ne savez pas comment faire.
Nous allons découvrir dans cette annexe tout ce qu'il faut savoir pour envoyer son site sur le Web :

1. Comment réserver un nom de domaine ?
2. Qu'est-ce qu'un hébergeur et comment cela fonctionne-t-il ?
3. Enfin, comment utiliser un client FTP pour pouvoir transférer les fichiers sur le Net ?

Le nom de domaine

Savez-vous ce qu'est un **nom de domaine** ?
Il s'agit en fait d'une adresse sur le Web : nordeclair.be est par exemple un nom de domaine.

Un nom de domaine est constitué de deux parties, ici "nordeclair" et ".be".

- Dans le cas présent, "noreclair" est le nom de domaine proprement dit. Il s'agit d'un nom que l'on peut en général choisir librement, tant que personne ne l'a réservé avant nous. Il peut contenir des lettres et des chiffres, et depuis 2012, certains caractères accentués (comme le « ç » français, le « é » ou le « è »).
- Le ".be" est l'extension (aussi appelée « TLD », de l'anglais *top-level domain*). Il existe grosso modo une extension par pays (.fr pour la France, .be pour la Belgique, .ca pour le Canada). Toutefois, il y a aussi des extensions utilisées au niveau international comme .com, .net, .org. Elles étaient au départ réservées aux sites commerciaux, aux organisations, … mais cela fait longtemps que tout le monde peut les réserver. D'ailleurs, .com est très probablement l'extension la plus utilisée sur le Web.

NB : En général, un site web voit son adresse précédée par www, comme par exemple www.lemonde.fr. Cela ne fait pas partie du nom de domaine : en fait, www est ce qu'on appelle un sous-domaine, et on peut en théorie en créer autant qu'on veut une fois qu'on est propriétaire du nom de domaine.
Le www est présent sur la plupart des sites web, c'est une sorte de convention, mais elle n'est absolument pas obligatoire.

Comment réserver son nom de domaine.

Pour réserver un nom de domaine, deux solutions :

- Passer par un registrar spécialisé. C'est un organisme qui sert d'intermédiaire entre l'ICANN (l'organisation qui gère l'ensemble des noms de domaine au niveau international) et vous. 1&1, OVH et Gandi sont de célèbres registrars français.
- Encore mieux : vous pouvez commander le nom de domaine en même temps que l'hébergement (c'est ce que je vous conseille). De cette manière, vous faites d'une pierre deux coups, vu que vous aurez de toute façon besoin de l'hébergement et du nom de domaine.

Quand tu crées un site chez par exemple Jimdo, Wix, 1&1, etc une version gratuite de ton nom de domaine et de son hébergement existe. Mais tu peux payer par an une modique somme afin d'avoir plus d'espace, plus de choix dans des modèles, plus de polices, etc.

Le code que tu viens d'apprendre permettra d'améliorer ton site bien que beaucoup d'options te seront proposées gratuitement.

L'hébergeur

Intéressons-nous maintenant à l'hébergeur.

Sur Internet, tous les sites web sont stockés sur des ordinateurs particuliers appelés **serveurs**. Ce sont des ordinateurs généralement très puissants, qui restent tout le temps allumés. Ils contiennent les pages des sites web et les délivrent aux internautes qui les demandent, à toute heure du jour et de la nuit.

L'hébergeur est une entreprise qui se charge de gérer des baies de serveurs. Elle s'assure du bon fonctionnement des serveurs 24h/24, 7j/7. En effet, si l'un d'eux tombe en panne, tous les sites présents sur la machine deviennent inaccessibles (et cela fait des clients mécontents).

Ces baies se situent dans des lieux particuliers appelés datacenters. Les datacenters sont donc en quelque sorte des « entrepôts à serveurs » et leur accès est très protégé.

Les hébergeurs, contrairement aux registrars, sont très très nombreux. Il y en a de tous types, à tous les prix. Il y a un vocabulaire à connaître pour vous repérer dans leurs offres :

- **Hébergement mutualisé** : si vous optez pour une offre d'hébergement mutualisé, votre site sera placé sur un serveur gérant plusieurs sites à la fois (peut-être une centaine, peut-être plus). C'est l'offre la moins chère et c'est celle que je vous recommande de viser si vous démarrez votre site web.
- **Hébergement dédié virtuel** : cette fois, le serveur ne gère que très peu de sites (généralement moins d'une dizaine). Cette offre est généralement adaptée aux sites qui d'un côté ne peuvent plus tenir sur un hébergement mutualisé car ils ont trop de trafic (trop de visiteurs), mais qui par ailleurs ne peuvent pas se payer un hébergement dédié (voir ci-dessous).
- **Hébergement dédié** (on parle aussi de « serveur dédié ») : c'est le nec plus ultra. Le serveur gère uniquement votre site et aucun autre. Attention, cela coûte assez cher et il vaut mieux avoir des connaissances en Linux pour administrer le serveur à distance.
- **Hébergement cloud** : de plus en plus en vogue, cela consiste à envoyer notre site sur des serveurs virtuels. En fait, c'est l'équivalent d'un hébergement dédié virtuel, mais avec tout un tas de services autour pour nous permettre de gérer plus facilement le réseau, les bases de données, etc. C'est la tendance pour de plus en plus de moyens et gros sites. Parmi les hébergeurs cloud, on peut citer Amazon Web Services, Google Cloud, Microsoft Azure, etc.
  Ce type d'hébergement est en revanche un peu trop complexe pour nous qui débutons dans la création de sites web. Je recommande plutôt un hébergement mutualisé dans notre cas.

Utiliser un client FTP

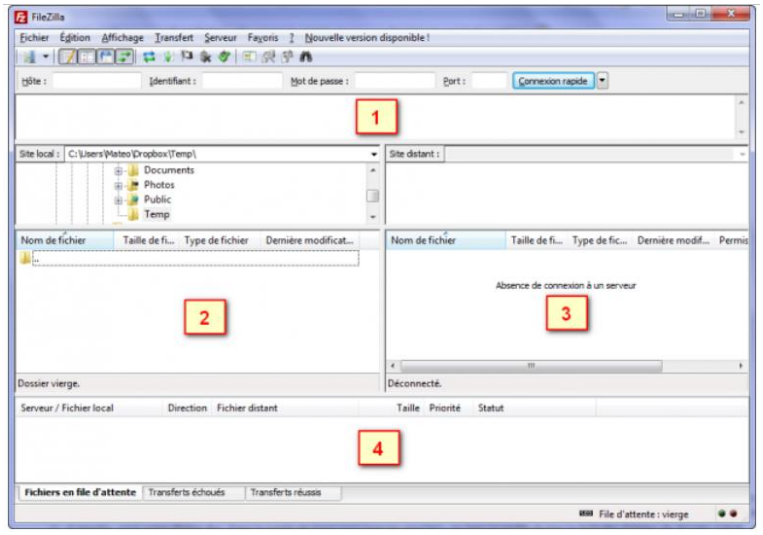
FTP signifie *File Transfer Protocol* et, pour faire court et simple, c'est le moyen que l'on utilise pour envoyer nos fichiers.
Il existe des logiciels permettant d'utiliser le FTP pour transférer vos fichiers sur Internet.

Bien entendu, des logiciels FTP, il en existe des centaines, gratuits, payants, français, anglais, etc.
Pour que nous soyons sur la même longueur d'onde, je vais vous proposer celui que j'utilise, qui est gratuit et en français : **FileZilla** (figure suivante).

Télécharger FileZilla

Prend la version correspondant à ton système d'exploitation (Windows, Mac OS X ou Linux).

Installe et lance le logiciel téléchargé.



À première vue, cela semble un peu compliqué (à première vue seulement). En fait, le principe est très simple.
Il y a quatre grandes zones à connaître dans la fenêtre :

1. En haut, tu verras apparaître les messages qu'envoie et reçoit le logiciel. Si tu as un peu de chance, tu verras même la machine te dire bonjour. En général, cette zone ne nous intéresse pas vraiment, sauf s'il y a des messages d'erreur en rouge…

2. À gauche, c'est ton disque dur. Dans la partie du haut, tu as les dossiers et, dans la partie du bas, la liste des fichiers du dossier actuel.
3. À droite, c'est la liste des fichiers envoyés sur le serveur sur Internet. Pour le moment il n'y a rien car on ne s'est pas connecté, mais cela va venir, aucune inquiétude.
4. Enfin, en bas, tu verras apparaître les fichiers en cours d'envoi (et le pourcentage d'envoi).

La première étape va être de se connecter au serveur de votre hébergeur.

Configurer le client FTP

Quel que soit l'hébergeur que tu as choisi, cela fonctionne toujours de la même manière. On va te fournir *trois informations* qui sont indispensables pour que FileZilla puisse se connecter au serveur :

- **L'IP** : c'est « l'adresse » du serveur. Le plus souvent, on te donnera une information du type `ftp.mon-site.com`, mais il peut aussi s'agir d'une suite de nombres comme 122.65.203.27.
- **Le login** : c'est ton identifiant, on t'a probablement demandé de le choisir. Tu as peut-être mis ton pseudo, ou le nom de ton site. Mon login pourrait par exemple être `mateo21`.
- **Le mot de passe** : soit on t'a demandé de choisir un mot de passe, soit (c'est plus probable) on t'en a attribué un d'office (un truc imprononçable du genre `crf45u7h`).

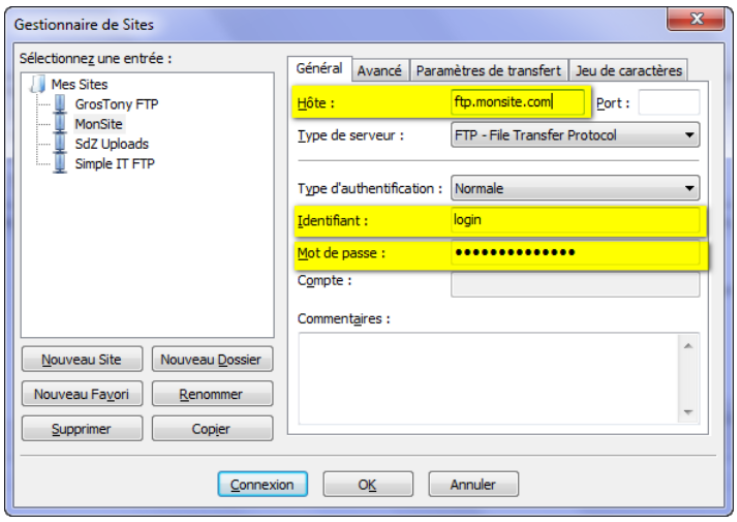
Si tu as ces trois informations, vous allez pouvoir continuer.
Si tu ne les as pas, il faut que tu les cherches, c'est indispensable. On te les a probablement envoyées par e-mail. Sinon, n'hésite pas à les demander à ton hébergeur (IP, login et mot de passe).

Maintenant que nous sommes en possession de ces informations, nous allons les donner à FileZilla, qui en a besoin pour se connecter au serveur.

Cliquez sur la petite icône en haut à gauche (pas sur la petite flèche à droite, mais bien sur l'image), représentée à la figure suivante.

L'icône de connexion de FileZilla

Une fenêtre s'ouvre. Cliquez sur Nouveau site et donne-lui le nom que tu veux (par exemple « Mon site »). À droite, tu vas devoir indiquer les trois informations dont je viens de vous parler, comme à la figure suivante.



Les trois informations à donner à FileZilla

Tu peux distinguer en haut l'hôte (c'est là qu'il faut indiquer ftp.monsite.com, par exemple). Coche Type d'authentification : Normale pour pouvoir saisir le login et le mot de passe.

Cliquez sur Connexion et le tour est (presque) joué.

Transférer les fichiers

À ce stade, deux possibilités :

- Soit la connexion a réussi : tu vas alors apparaître en haut des messages en vert comme « Connecté ». Dans ce cas, la zone de droite de la fenêtre de FileZilla devrait s'activer et vous verrez les fichiers qui se trouvent déjà sur le serveur (il se peut qu'il y en ait déjà quelques-uns).
- Soit cela a planté, tu as plein de messages écrits en rouge et là, eh bien… il n'y a pas trente-six solutions : tu t'es trompé en tapant l'IP, ou le login, ou le mot de passe. Un de ces éléments est incorrect, veille à les redemander à ton hébergeur car s'ils sont bons cela *doit* marcher.

Si la connexion a réussi, alors ce que tu as à faire est très simple : dans la partie de gauche, cherches où se trouvent, sur ton disque dur, ton fichiers .html et .css (mais aussi les images.jpg,.png,.gif, etc.).
À gauche, fais un double-clic sur le fichier que tu vas transférer. Au bout de

quelques secondes, il apparaîtra à droite, ce qui voudra dire qu'il a été correctement envoyé sur le serveur, et donc qu'il est accessible sur Internet !

Tu peux envoyer n'importe quel type de fichier. Bien entendu, généralement on envoie des fichiers `.php`, `.html`, `.css` et des images, mais tu peux aussi très bien envoyer des `.pdf`, des programmes, des `.zip`, etc.

À noter qu'il *faut* que la page d'accueil s'appelle index.html. C'est la page qui sera chargée lorsqu'un nouveau visiteur arrivera sur ton site.

En résumé

- Pour le moment, ton site web n'est visible que par toi, sur ton ordinateur. Il faut l'envoyer sur le Web pour qu'il soit visible par tout le monde.
- Tu as besoin de deux éléments :
    - Un nom de domaine : c'est l'adresse de ton site web. Tu peux réserver une adresse en .com, .fr, .net… Par exemple : dhnet.be.
    - Un hébergeur : c'est lui qui va stocker ton site web sur une machine appelée « serveur ». Son rôle sera d'envoyer ton site à tes visiteurs à toute heure du jour et de la nuit.
- Pour transmettre les fichiers de ton site au serveur de ton hébergeur, il faut utiliser un client FTP comme FileZilla.
- Pour te connecter au serveur, tu as besoin de trois informations : l'adresse IP du serveur (ou son nom d'hôte), ton login et ton mot de passe. Ceux-ci vous sont fournis par ton hébergeur.

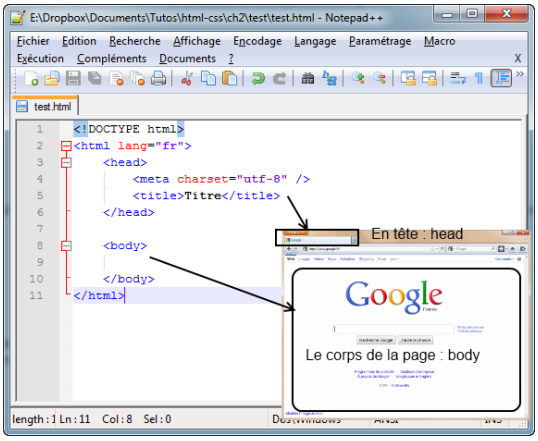
Chapitre 12 : Mémento des balises HTML

Tu trouveras ici un grand nombre de balises HTML

Tu peux te servir de cette annexe comme d'un aide-mémoire lorsque tu développes ton site web.

Balise	Description
<html>	Balise principale
<head>	En-tête de la page
<body>	Corps de la page

Code minimal d'une page Web



Balises d'en-tête

Ces balises sont toutes situées dans l'en-tête de la page web, c'est-à-dire entre <head> et </head> :

Balise	Description
`<link />`	Liaison avec une feuille de style
`<meta />`	Métadonnées de la page web (charset, mots-clés, etc.)
`<script>`	Code JavaScript
`<style>`	Code CSS
`<title>`	Titre de la page

Balises de structuration du texte

Balise	Description
`<abbr>`	Abréviation
`<blockquote>`	Citation (longue)
`<cite>`	Citation du titre d'une œuvre ou d'un évènement
`<q>`	Citation (courte)
`<sup>`	Exposant
`<sub>`	Indice
`<strong>`	Mise en valeur forte
`<em>`	Mise en valeur normale
`<mark>`	Mise en valeur visuelle
`<h1>`	Titre de niveau 1
`<h2>`	Titre de niveau 2
`<h3>`	Titre de niveau 3
`<h4>`	Titre de niveau 4
`<h5>`	Titre de niveau 5
`<h6>`	Titre de niveau 6

`<img />`	Image
`<figure>`	Figure (image, code, etc.)
`<figcaption>`	Description de la figure
`<audio>`	Son
`<video>`	Vidéo
`<source>`	Format source pour les balises `<audio>` et `<video>`
`<a>`	Lien hypertexte
` `	Retour à la ligne
`<p>`	Paragraphe
`<hr />`	Ligne de séparation horizontale
`<address>`	Adresse de contact
`<del>`	Texte supprimé
`<ins>`	Texte inséré
`<dfn>`	Définition
`<kbd>`	Saisie clavier
`<pre>`	Affichage formaté (pour les codes sources)
`<progress>`	Barre de progression
`<time>`	Date ou heure

Balises de listes

Cette section énumère toutes les balises HTML permettant de créer des listes (listes à puces, listes numérotées, listes de définitions…)

Balise	Description
`<ul>`	Liste à puces, non numérotée
`<ol>`	Liste numérotée
`<li>`	Élément de la liste à puces
`<dl>`	Liste de définitions
`<dt>`	Terme à définir
`<dd>`	Définition du terme

Balises de tableau

Balise	Description
`<table>`	Tableau
`<caption>`	Titre du tableau
`<tr>`	Ligne de tableau
`<th>`	Cellule d'en-tête
`<td>`	Cellule
`<thead>`	Section de l'en-tête du tableau
`<tbody>`	Section du corps du tableau
`<tfoot>`	Section du pied du tableau

Balises de formulaire

Balise	Description
`<form>`	Formulaire
`<fieldset>`	Groupe de champs
`<legend>`	Titre d'un groupe de champs
`<label>`	Libellé d'un champ
`<input />`	Champ de formulaire (texte, mot de passe, case à cocher, bouton, etc.)
`<textarea>`	Zone de saisie multiligne
`<select>`	Liste déroulante
`<option>`	Élément d'une liste déroulante
`<optgroup>`	Groupe d'éléments d'une liste déroulante

Balises sectionnantes

Ces balises permettent de construire le squelette de notre site web.

Balise	Description
`<header>`	En-tête
`<nav>`	Liens principaux de navigation
`<footer>`	Pied de page
`<section>`	Section de page
`<article>`	Article (contenu autonome)
`<aside>`	Informations complémentaires

Balises génériques

Les balises génériques sont des balises qui n'ont pas de sens sémantique.

En effet, toutes les autres balises HTML ont un *sens* : <p> signifie « Paragraphe », <h2> signifie « Sous-titre », etc.
Parfois, on a besoin d'utiliser des balises génériques (aussi appelées **balises universelles**) car aucune des autres balises ne convient. On utilise le plus souvent des balises génériques pour construire son design.

Il y a deux balises génériques : l'une est `inline`, l'autre est `block`.

Balise	Description
`<span>`	Balise générique de type inline
`<div>`	Balise générique de type block

Ces balises ont un intérêt uniquement si tu leur associes un attribut `class`, `id` ou `style` :

- `class`: indique le nom de la classe CSS à utiliser.
- `id`: donne un nom à la balise. Ce nom doit être unique sur toute la page car il permet d'identifier la balise. Tu peux te servir de l'ID pour de nombreuses choses, par exemple pour créer un lien vers une ancre, pour un style CSS de type ID, pour des manipulations en JavaScript, etc.
- `style`: cet attribut te permet d'indiquer directement le code CSS à appliquer. Tu n'es donc pas obligé d'avoir une feuille de style à part, tu peux mettre directement les attributs CSS. Notes qu'il est préférable de ne pas utiliser cet attribut et de passer à la place par une feuille de style externe, car cela rend ton site plus facile à mettre à jour par la suite.

Ces trois attributs ne sont pas réservés aux balises génériques : tu peux aussi les utiliser sans aucun problème dans la plupart des autres balises.

Chapitre 13 : Mémento des propriétés CSS

Tu trouveras ici un grand nombre de propriétés CSS

Tu peux te servir de cette annexe comme d'un aide-mémoire lorsque tu développes ton site web.

Propriétés de mise en forme du texte

Propriété	Valeurs (exemples)	Description
`font-family`	*police1, police2, police3*, serif, sans-serif, monospace	Nom de police
`@font-face`	*Nom et source de la police*	Police personnalisée
`font-size`	1.3em, 16px, 120%...	Taille du texte
`font-weight`	bold, normal	Gras
`font-style`	italic, oblique, normal	Italique
`text-decoration`	underline, overline, line-through, blink, none	Soulignement, ligne au-dessus, barré ou clignotant
`font-variant`	small-caps, normal	Petites capitales
`text-transform`	capitalize, lowercase, uppercase	Capitales
`font`	-	Super propriété de police. Combine : `font-weight`, `font-style`, `font-size`, `font-variant`, `font-family`.
`text-align`	left, center, right, justify	Alignement horizontal
`vertical-align`	baseline, middle, sub, super, top, bottom	Alignement vertical (cellules de tableau ou éléments `inline-block` uniquement)

`line-height`	18px, 120%, normal...	Hauteur de ligne
`text-indent`	25px	Alinéa
`white-space`	pre, nowrap, normal	Césure
`word-wrap`	break-word, normal	Césure forcée
`text-shadow`	5px 5px 2px blue *(horizontale, verticale, fondu, couleur)*	Ombre de texte

Propriétés de couleur et de fond

Propriété	Valeurs (exemples)	Description
`color`	*nom*, rgb(rouge,vert,bleu), rgba(rouge,vert,bleu,transparence), #CF1A20...	Couleur du texte
`background-color`	*Identique à color*	Couleur de fond
`background-image`	url('image.png')	Image de fond
`background-attachment`	fixed, scroll	Fond fixe
`background-repeat`	repeat-x, repeat-y, no-repeat, repeat	Répétition du fond
`background-position`	*(x y)*, top, center, bottom, left, right	Position du fond
`background`	-	Super propriété du fond. Combine : `background-image` , `background-repeat` , `background-attachment` , `background-position`
`opacity`	0.5	Transparence

Propriétés des boîtes

Propriété	Valeurs (exemples)	Description
`width`	150px, 80%...	Largeur
`height`	150px, 80%...	Hauteur
`min-width`	150px, 80%...	Largeur minimale
`max-width`	150px, 80%...	Largeur maximale
`min-height`	150px, 80%...	Hauteur minimale
`max-height`	150px, 80%...	Hauteur maximale
`margin-top`	23px	Marge en haut
`margin-left`	23px	Marge à gauche
`margin-right`	23px	Marge à droite
`margin-bottom`	23px	Marge en bas
`margin`	23px 5px 23px 5px *(haut, droite, bas, gauche)*	Super-propriété de marge. Combine : `margin-top`, `margin-right`, `margin-bottom`, `margin-left`.
`padding-left`	23px	Marge intérieure à gauche

`padding-right`	23px	Marge intérieure à droite
`padding-bottom`	23px	Marge intérieure en bas
`padding-top`	23px	Marge intérieure en haut
`padding`	23px 5px 23px 5px *(haut, droite, bas, gauche)*	Super-propriété de marge intérieure. Combine : `padding-top` , `padding-right` , `padding-bottom` , `padding-left` .
`border-width`	3px	Épaisseur de bordure
`border-color`	*nom*, rgb(rouge,vert,bleu), rgba(rouge,vert,bleu,transparence), #CF1A20...	Couleur de bordure
`border-style`	solid, dotted, dashed, double, groove, ridge, inset, outset	Type de bordure
`border`	3px solid black	Super-propriété de bordure. Combine `border-width` , `border-color` , `border-style` . Existe aussi en version `border-top` , `border-right` , `border-bottom` , `border-left` .
`border-radius`	5px	Bordure arrondie
`box-shadow`	6px 6px 0px black *(horizontale, verticale, fondu, couleur)*	Ombre de boîte

Propriétés de positionnement et d'affichage

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

Propriétés des listes

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

Propriétés des tableaux

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

Autres propriétés

Propriété	Valeurs (exemple)	Description
`cursor`	crosshair, default, help, move, pointer, progress, text, wait, e-resize, ne-resize, auto...	Curseur de souris

Bibliographie

- Site Open Class Room

- Réalisez votre site Web avec l'HTML 5 – Mathieu Nebra – Le Livre du zéro

- HTML5 et CSS3 : Maîtrisez les standards des applications Web 2ème Edition – Luc Van Lancker – Eni Edition