



HTML & CSS

Définitions:

Hypertext Markup Language ⇒ créer une page web(texte, image...)

Une page est constituée d'éléments HTML(balises et contenu).

Cascading Style sheets pour le changement de couleur, position, police.....

Doctype indique qu'on va utiliser de l'html5 ou 6..

-head c'est là où on met les métadonnées.

Pour préparer l'environnement de travail, on installe live server pour un service en temps réel.

CSS(cascading style sheets): langage de description pour gérer l'apparence de la page web.

Navigateur envoie la requête au serveur qui contient les infos sur le site et renvoie ça au navigateur qui va l'interpréter et l'afficher.

- Le HTML constitue la structure d'une page web.
- Le CSS permet d'ajouter du style.
- Les deux langages se complètent avec un rôle bien défini pour chacun.
- Le navigateur est un logiciel qui permet de lire les langages du Web : HTML et CSS.
- Tous les navigateurs embarquent des outils de développement, dont l'outil d'inspection qui permet d'accéder au HTML et au CSS d'une page.

Balises fermantes(balises en paires):

ul/ol liste ordonnée et non ordonnée avec des li(list item)dedans .

<p>:paragraphe ;

<h1.....6>: pour l'ordre du texte et des différents titres.

<div>: pour diviser

lorem: pour créer du texte aléatoire.

 : pour identifier une petite partie du texte. c'est à dire isoler.

Balises non fermantes(appelés balises orphelines):

 break pour aller à la ligne.

<hr>: créer ligne horizontale.

 ayant pour attribut **src=""** pour le chemin d'une image; **alt=""** pour le texte descriptif de l'image.

Commentaire: <!-- -->

Raccourci pour commenter: ctrl + /

- La balise en paire `<head> </head>` contient deux balises qui donnent des informations au navigateur : l'encodage et le titre de la page.
- La balise orpheline `<meta charset="utf-8">` indique l'encodage utilisé dans le fichier `.html` : cela détermine comment les caractères spéciaux s'affichent (accents, idéogrammes chinois et japonais, etc.).
- `<html lang="fr">` pour le français ;
- `<html lang="en">` pour l'anglais ;
- `<html lang="es">` pour l'espagnol...

Les balises pour mettre en valeur du texte:

`<small>` pour rendre le texte plus petit

``: pour mettre en gras (mettre en importance plutôt)

``: pour mettre en italique

`<mark></mark>`: surligner du texte.

En fait `strong` et `em` sont intéressants car les robots des moteurs de recherche parcourent le web en lisant le code html.

Les liens hypertextes:

`<a>` (pour anchor) "" avec pour attribut `href=""` dans lequel on indique le lien de la page vers laquelle on va.

. lien relatif (indique où trouver notre fichier html) ou absolue (indique une adresse complète).

Création d'ancre.

```
<balise id="haut"> </balise>
```

```
<a href="#haut"> : ça c'est pour se mouvoir dans
```

2ème cas: ancre située sur une autre page.

on fait: ``

donc ``

Pour faire de sorte qu'il s'ouvre dans un autre onglet, on fait:

```
<a href="" target="_blank">
```

- `href="mailto:NOMDUMAIL@MAIL.COM"` crée un lien hypertexte qui ouvre la boîte mail avec un nouveau message vide.
- `href="NOMDEFICHIER.EXTENSION"` crée un lien hypertexte qui permet de télécharger un fichier que vous avez placé au préalable dans le même dossier que votre page web.

Si votre fichier cible est placé dans un dossier qui se trouve "plus haut" dans l'arborescence, il faut écrire deux points `..` , comme ceci :

```
<a href=" ../page3.html">Page 3</a>
```

Les images:

- Il existe plusieurs formats d'images adaptés au Web : PNG, JPG...
- On insère une image avec la balise `` .
- `` doit obligatoirement comporter au moins ces deux attributs : `src` (source de l'image) et `alt` (courte description de l'image).
- **!** On peut ajouter l'attribut `title` " " à image pour afficher une info-bulle
- Image cliquable: `<a>`

CSS:

- CSS est un autre langage qui vient compléter le HTML. Son rôle est de mettre en forme votre page web.
- Pour écrire le code CSS, on crée un fichier séparé portant l'extension `.css` comme `style.css` .
- Pour lier les fichiers CSS et HTML, on rajoute une ligne dans la balise `<head>` `</head>` du fichier HTML : `<link href="style.css" rel="stylesheet">`
- En CSS, on sélectionne les portions de la page HTML qu'on veut modifier, et on change leur présentation avec des propriétés CSS :

```
balise1
{
    propriete1: valeur1;
```

```
    propriete2: valeur2;
}
```

Donner le même attribut à plusieurs balises: :

```
h1, p{
}
```

Les attributs `class` et `id` fonctionnent selon la même méthode mais on ne les utilise pas pour les mêmes raisons :

L'attribut `id` fonctionne selon la même méthode que `class` , mais il y a une différence de taille : `id` ne peut être utilisé **qu'une fois** dans le code.

En CSS, on peut appliquer du style à un élément (ou plus) avec l'attribut `class` . Par contre `id` ne peut s'utiliser que pour un seul élément, pas plus.

Pour appeler une classe en css: (précédé d'un point)

```
.ma-classe {
color: #663399;
}
```

appeler id(précédé de #):

```
#logo {
/* Indiquez les propriétés CSS ici */
}
```

Exploiter les balises universelles:

En fait, on a inventé deux balises dites "universelles", qui n'ont aucune signification particulière (elles n'indiquent pas que le mot est important, par

exemple). Il y a une petite différence (mais significative) entre ces deux balises :

1. ` `: balise de type inline c'est-à-dire que l'on place au sein d'un paragraphe pour sélectionner certains mots seulement
2. `<div> </div>` : balise block qui encadre un bloc de texte.

Les balises `<p>` , `<h1>` , etc., sont de la même famille. Ces balises ont quelque chose en commun : elles créent un nouveau "bloc", dans la page, et provoquent donc obligatoirement un retour à la ligne. `<div>` est une balise fréquemment utilisée dans la construction d'une mise en page, comme nous le verrons plus tard.

Propriétés CSS intéressantes:

FONT-SIZE:

- On modifie la taille du texte avec la propriété CSS `font-size`: on indique une taille absolue (précis mais pas responsive => du pixel: `px`) ou une taille relative (plus souple => `em` (recommandé))

Attention : les unités sont collées aux nombres.

- On peut indiquer la taille en pixels, comme `16px` ; ou encore en "em", comme `1.3em` .
- On indique la police du texte avec la propriété CSS `font-family` .
- De nombreuses propriétés de mise en forme du texte existent : `font-style` pour l'italique, `font-weight` pour la mise en gras, `text-decoration` pour le soulignement.
- Le texte peut être aligné avec la propriété CSS `text-align` .

FONT-FAMILY: pour indiquer la police

Liste de police sans sérif:

- Arial Black ;
- Futura ;
- Helvetica ;

- Impact ;
- Trebuchet MS ;
- Verdana.



Pour avoir n'importe lequel des polices qu'on veut: aller sur

Browse Fonts - Google Fonts

Making the web more beautiful, fast, and open through great typography

 <https://fonts.google.com/>



, puis :

1. Copiez les balises `<link>` dans le `<head> </head>` du fichier HTML.
2. Utilisez la propriété `font-family` dans le fichier CSS pour déclarer que vous voulez utiliser cette police.

Par exemple, pour la police Roboto, on vient coller dans le HTML :

```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
```

```
<link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap" rel="stylesheet">
```

Et on l'utilise dans le CSS en déclarant dans notre sélecteur :

```
font-family: 'Roboto', sans-serif;
```

FONT-style:

- `italic` : le texte sera mis en italique ;
- `normal` : le texte sera normal (par défaut). Cela vous permet d'annuler une mise en italique

FONT-WEIGHT:

bold: gras;

normal: normalement;

thin: plus fin.

mais pour être plus précis on peut juste mettre des valeurs:

Thin (100)
Extra-Light (200)
Light (300)
Regular (400)
Medium (500)
SemiBold (600)
Bold (700)
ExtraBold / Heavy (800)
Black (900)

Pour appliquer les différents styles de texte (épaisseur et italique) pour les polices importées, il faut bien s'assurer d'avoir importé les styles de polices correspondants. Ainsi, pour utiliser la police Roboto en italique et bold, il faudra bien avoir importé dans votre code :

```
<link href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@1,700&display=swap" rel="stylesheet">
```

Text-decoration:

underline: souligné;

line-through: barré;

none: normal

text-align:

left; right; center, justify.

justifier le texte permet de faire en sorte qu'il prenne toute la largeur possible sans laisser d'espace blanc à la fin des lignes. Les textes des journaux, par exemple, sont toujours justifiés.

Couleur de font:

site pour facilement choisir les couleurs dans le format désiré:

https://www.w3schools.com/colors/colors_picker.asp

- On change la couleur du texte avec la propriété `color` et la couleur de fond avec la propriété `background-color`.
- On peut indiquer une couleur en écrivant son nom en anglais, `black` par exemple, sous forme hexadécimale, comme `#FFC8D3`, ou en notation RGB, comme `rgb(250, 25, 118)`.
- On peut ajouter une image de fond avec la propriété `background-image`. On peut choisir de fixer l'image de fond, ou encore de la positionner où on veut

sur la page.

Modifiez le comportement d'une image de fond

Pour changer le comportement d'une image de fond, il existe plusieurs propriétés CSS :

1. La propriété CSS `background-attachment` associée à la valeur `fixed` permet de rendre l'image de fond fixe lorsqu'on déroule la page web : `background-attachment: fixed;`
2. La propriété CSS `background-size` associée à la valeur `cover` permet de redimensionner l'image afin qu'elle s'adapte à la taille de l'élément qui la contient (elle garde ses proportions, en rognant la largeur ou la hauteur en fonction de la taille de l'élément qui la contient) : `background-size: cover;`
3. La propriété CSS `background-position` associée aux valeurs `top` , `bottom` , `left` , `center` ou `right` permet d'indiquer où doit se trouver l'image de fond, par exemple : `background-position: top right;`
- On peut rendre une portion de la page transparente avec la propriété `opacity` ou avec la notation `RGBA` (une extension de la notation RGB, où la quatrième valeur indique le niveau de transparence).

Combinez ces propriétés CSS avec la "super-propriété" `background`

Si vous utilisez beaucoup de propriétés en rapport avec le fond, vous pouvez utiliser une sorte de "super-propriété" appelée `background` dont la valeur peut combiner plusieurs des propriétés :

- `background-image` ;
- `background-repeat` ;
- `background-attachment` ;
- `background-size` ;
- et `background-position` .

C'est la première "super-propriété" que je vous montre, il y en aura d'autres.

On peut donc tout simplement écrire :

```
.banniere
{
```

```
background: url("paysage.jpg") cover center;
}
```

Créez des dégradés avec **linear-gradient**

Pour créer un dégradé, on a besoin de la propriété CSS **background** :

```
background: linear-gradient(90deg, #8360c3, #2ebf91);
```

Si je devais lire en français cette ligne de CSS, voici ce que ça donnerait :

"J'applique un dégradé linéaire, à 90 degrés, en partant de la couleur #8360c3 pour arriver à la couleur #2ebf91.

Jouez sur la transparence avec la propriété CSS **opacity**

La propriété CSS **opacity** permet d'indiquer le niveau d'opacité (c'est l'inverse de la transparence).

- Avec une valeur de 1, l'élément sera totalement opaque : c'est le comportement par défaut.
- Avec une valeur de 0, l'élément sera totalement transparent.

Il faut donc choisir une valeur comprise entre 0 et 1. Ainsi, avec une valeur de 0.6 , votre élément sera opaque à 60 %... et on verra donc à travers !

En CSS, si vous appliquez un style à une balise, toutes les balises qui se trouvent à l'intérieur prendront le même style. Cela s'appelle **l'héritage** : on dit que les balises qui se trouvent à l'intérieur d'une autre balise "héritent" de ses propriétés. (L'héritage ne fonctionne pas uniquement pour la couleur, mais pour toutes les propriétés CSS.)

BORDURE en CSS:

- On peut appliquer une bordure à un élément avec la super-propriété CSS `border`. Il faut indiquer la largeur de la bordure, sa couleur et son type (simple, double, pointillés, tirets).
1. **La largeur** que l'on définit avec une valeur en pixels (comme 2px).
 2. **La couleur** que l'on indique avec un nom de couleur, une valeur hexadécimale, ou une valeur RGB.
 3. **Le type de bordure** qui peut être `solid` (un trait simple), `double` (un double trait), `dotted` (un trait en pointillés), `dashed` (un trait en tirets), ou autre. Vous avez un large panel d'options :

vous voulez mettre des bordures différentes en fonction du côté (haut, bas, gauche ou droite), vous pouvez le faire sans problème. Dans ce cas, vous devrez utiliser ces quatre propriétés :

1. `border-top` : bordure du haut.
2. `border-bottom` : bordure du bas.
3. `border-left` : bordure de gauche.
4. `border-right` : bordure de droite.

Il existe aussi des équivalents pour paramétrer chaque détail de la bordure si vous le désirez :

- `border-top-width` pour modifier l'épaisseur de la bordure du haut,
- `border-top-color` pour la couleur du haut, etc.

- On peut arrondir les bordures avec la propriété CSS `border-radius`.


Dans ce cas, indiquez les quatre valeurs correspondant aux angles dans la propriété `border-radius`, dans cet ordre :

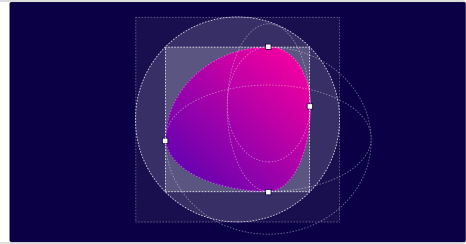
1. En haut à gauche.
2. En haut à droite.
3. En bas à droite.
4. En bas à gauche.

Astuce pour trouver les bonnes valeurs selon la forme désirée:

Fancy Border Radius Generator

Generator to build organic shapes with CSS3 border-radius

 <https://9elements.github.io/fancy-border-radius/>



- On peut ajouter une ombre aux blocs de texte avec `box-shadow`. On doit indiquer le décalage vertical et horizontal de l'ombre, son niveau d'adoucissement et sa couleur.

exemple: `box-shadow: 6px 6px 0px rgba(0, 0, 0);`

la propriété CSS `box-shadow` s'applique à tout le bloc, et prend quatre valeurs dans l'ordre suivant :

1. Le décalage horizontal de l'ombre.
2. Le décalage vertical de l'ombre.
3. L'adoucissement du dégradé.
4. La couleur de l'ombre.

L'adoucissement peut être :

-

faible (si on lui donne une **valeur inférieure** à celle du décalage),

-

normal (si on lui donne une **valeur égale** à celle du décalage)


- ou

élevé (si on lui donne une **valeur supérieure** à celle du décalage).

Trouver un ombrage bien désiré:

Smooth Shadow

Make a smooth css shadow

 <https://shadows.brumm.af/>

- Le texte peut lui aussi avoir une ombre avec `text-shadow`.

STYLISER LE SURVOL:

- En CSS, on peut modifier l'apparence de certaines sections dynamiquement, après le chargement de la page, lorsque certaines interactions se produisent. On utilise pour cela les pseudo-classes.
- La pseudo-classe `:hover` modifie l'apparence d'un élément au survol (par exemple : `a:hover` modifie l'apparence des liens hypertextes lorsque la souris pointe dessus).
- La pseudo-classe `:active` modifie l'apparence des liens hypertextes au moment du clic.
- La pseudo-classe `:visited` modifie l'apparence des liens hypertextes lorsqu'un lien a déjà été visité.
- La pseudo-classe `:focus` modifie l'apparence d'un élément sélectionné via la touche "tab".
- Encore aujourd'hui, certaines propriétés ne sont pas totalement reconnues par tous les navigateurs.

Allez plus loin avec les sélecteurs avancés

En CSS, le plus dur est bien souvent de réussir à cibler l'élément dont on veut changer le style. Vous avez appris tout un tas de manières d'appliquer du style à des éléments, avec les sélecteurs et les pseudo-classes. Mais sachez que nous n'avons pas tout couvert dans cette partie. Vous avez eu un aperçu de la puissance des sélecteurs avec le combinateur de voisin direct (ou siblings) dans la vidéo d'introduction, mais il en existe d'autres. En voici quelques-uns :

Le sélecteur universel

Le sélecteur universel `*` sélectionne toutes les balises sans exception.

```
* {  
/* Insérez ici votre style */  
}
```

Le sélecteur d'une balise contenue dans une autre : **A** **B**

Prenons un exemple avec ce code écrit dans le fichier HTML :

```
<h3>Titre avec <em>texte important</em></h3>
```

Dans CSS, on écrirait alors :

```
h3 em {  
/* Insérez ici votre style */  
}
```

Ce morceau de code signifie en français :

"Applique ce style à toutes les balises `` situées à l'intérieur d'une balise `<h3>` ".

Notez qu'il n'y a pas de virgule entre les deux noms de balises.

Le sélecteur d'une balise qui en suit une autre : **A** + **B**

Dans CSS, on écrirait par exemple :

```
h3 + p {  
/* Insérez ici votre style */  
}
```

Ce qui aura pour résultat de sélectionner la première balise `<p>` située après un titre `<h3>`.

Exemple de code HTML associé :

```
<h3>Titre</h3>  
<p>Paragraphe</p>
```

Le sélecteur d'une balise qui possède un attribut : **a[attribut]**

```
a[title] {  
/* Insérez ici votre style */  
}
```

```
}
```

En français, ce morceau de code signifie :

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

Exemple de code HTML associé :

```
<a href="http://site.com" title="Infobulle">
```

Il existe des variantes de cette forme de sélecteur :

- `a[attribut="Valeur"]` : une balise qui possède un attribut et une valeur exacte, comme :

```
a[title="Cliquez ici"] {  
/* Insérez ici votre style */  
}
```

C'est la même chose, mais l'attribut doit en plus avoir exactement pour valeur "Cliquez ici".

Exemple de code HTML associé :

```
<a href="http://site.com" title="Cliquez ici">
```

- `a[attribut*="Valeur"]` : une balise, un attribut et une valeur, comme :

```
a[title*="ici"] {  
/* Insérez ici votre style */  
}
```

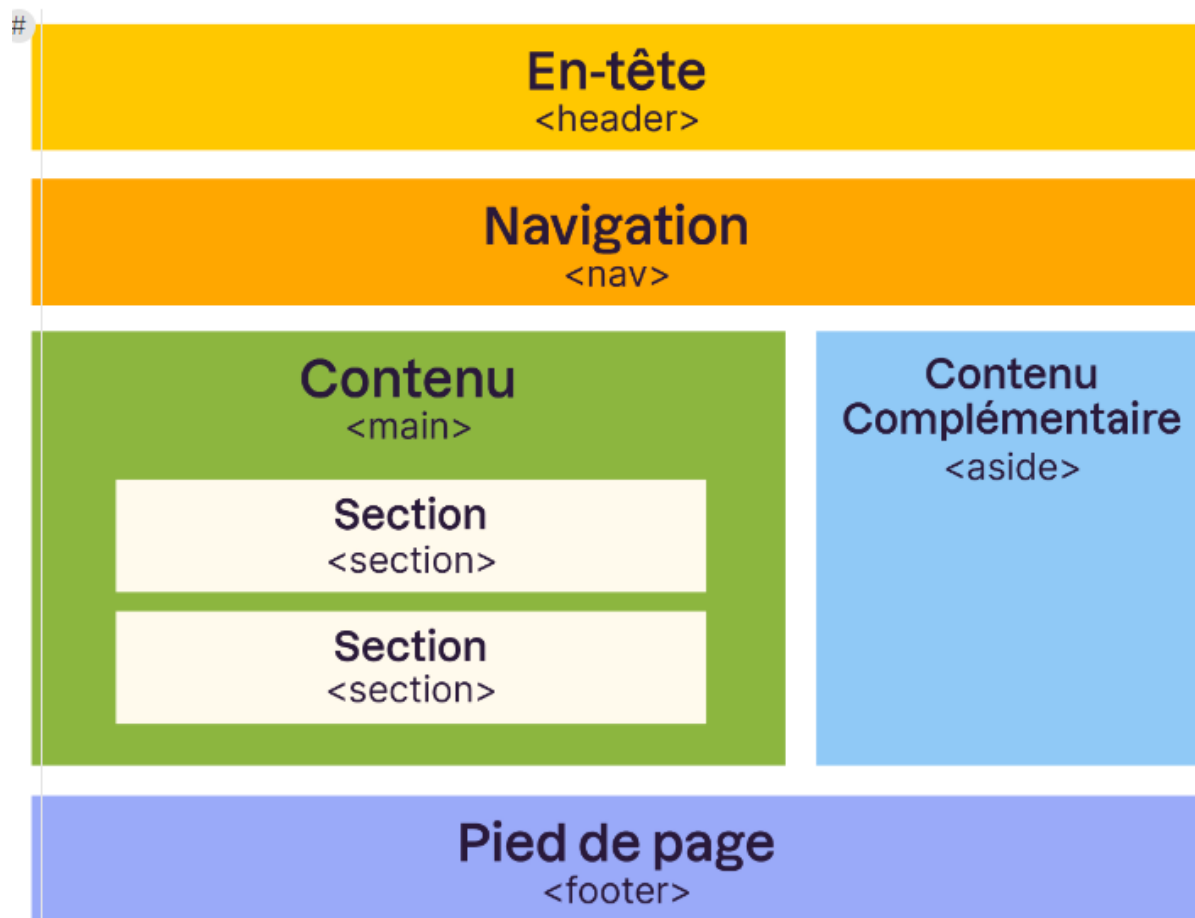
Idem, l'attribut doit cette fois contenir dans sa valeur le mot "ici" (peu importe sa position).

Exemple de code HTML associé :

```
<a href="http://site.com" title="Quelque part par ici">
```


STRUCTURATION DE LA PAGE:

- Plusieurs balises permettent de délimiter les différentes zones qui constituent la page web :
 - `<header>` : en-tête ;
 - `<footer>` : pied de page ;
 - `<nav>` : liens principaux de navigation ;
 - `<main>` : le contenu de la page principale de la page dans le quel on a les éléments qui suivent.
 - `<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 la fonction du texte qu'elles encadrent. On pourrait très bien placer l'en-tête en bas de la page, si on le souhaitait.



Différencier les balises de type `block` et de type `inline`

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

- `block` : ce type de balise crée automatiquement un retour à la ligne avant et après ;
- `inline` : ce type se trouve obligatoirement à l'intérieur d'une balise `block` .

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

1. `margin` (taille de la **marge extérieure**)
2. `padding` (taille de la **marge intérieure**).

exemple `margin: hauts droit bas gauche;` ; donc un cercle dans le sens d'une montre.

Spécifiez les propriétés `margin` et `padding`

Vous allez avoir besoin d'un minimum de vocabulaire en anglais ici :

- `top` : haut ;
- `bottom` : bas ;
- `left` : gauche ;
- `right` : droite.

Centrez vos blocs avec `width` et `margin: auto;`

Pour centrer des blocs, il faut respecter les règles suivantes :

1. donner une largeur au bloc avec la propriété `width` ;
2. indiquer `margin: auto;` (les marges extérieures seront alors automatiques, et permettront de centrer le contenu).

GROSSE GROSSE ASTUCE :

```
* {  
  margin: 0;  
}
```

Cela sert à réinitialiser les marges par défaut des navigateurs.

- On distingue deux principaux types de balises en HTML :
 - les balises de type `block` comme `<p>` ou `<h1>` créent un retour à la ligne et occupent par défaut toute la largeur disponible. Elles se suivent de haut en bas ;
 - les balises de type `inline` comme `<a>` ou `` 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).
- Les éléments de la page disposent chacun de `padding` (marges intérieures) et de `margin` (marges extérieures).

- On peut centrer le contenu d'un bloc dont la largeur est définie par `width` avec `margin: auto;`

FLEXBOX:

- Le principe de Flexbox est d'avoir un conteneur avec plusieurs éléments à l'intérieur comme un carton . 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*.

Flexdirection a comme

attribut: row

, column ,

row-reverse,

column-reverse

exemple:

on fait simplement :

```
.container {  
  display: flex;  
  flex-direction: column;  
}
```

Par défaut, les blocs essaient de rester sur la même ligne s'ils n'ont pas la place, quitte à "s'écraser", et provoquer parfois des anomalies dans la mise en page (certains éléments pouvant dépasser de leur conteneur). Si vous voulez, vous pouvez demander à ce que les blocs aillent à la ligne lorsqu'ils n'ont plus la place, avec `flex-wrap` .

Voilà les différentes valeurs de flex-wrap :

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

```
.container {  
display: flex;  
flex-wrap: nowrap;  
    /* OU wrap;  
    OU wrap-reverse; */  
}
```

- L'alignement des éléments se fait sur l'axe principal (horizontal si aligné horizontalement) avec `justify-content` avec comme attribut:
- `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.
- et sur l'axe secondaire avec `align-items` .(donc justify content pour l'axe principal et align-items pour gerer l'alignement secondaire.

et align-items a pour valeur:

- `stretch` : les éléments sont étirés sur tout l'axe (valeur par défaut) ;
- `flex-start` : alignés au début ;
- `flex-end` : alignés à la fin ;
- `center` : alignés au centre ;
- `baseline` : alignés sur la ligne de base (semblable à `flex-start`).

- 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`.

comme on est souvent en horizontale donc pour aligner les flexbox horizontalement, on fait `justify-content`, si verticalement; `align-items`.

Répartissez les blocs sur plusieurs lignes avec `align-content`

Si vous avez plusieurs lignes dans votre Flexbox, vous pouvez choisir comment celles-ci seront réparties avec `align-content`.

Cette propriété n'a aucun effet s'il n'y a qu'une seule ligne dans la Flexbox.

Prenons donc un cas de figure où nous avons plusieurs lignes. J'autorise les éléments à aller à la ligne avec `flex-wrap`.

Voyons voir comment les lignes se répartissent différemment avec la nouvelle propriété `align-content` que je voulais vous présenter. Elle peut prendre ces valeurs :

- `stretch` (par défaut) : les éléments s'étirent pour occuper tout l'espace ;
- `flex-start` : les éléments sont placés au début ;
- `flex-end` : les éléments sont placés à la fin ;
- `center` : les éléments sont placés au centre ;
- `space-between` : les éléments sont séparés avec de l'espace entre eux ;
- `space-around` : idem, mais il y a aussi de l'espace au début et à la fin.

DISPLAY:GRID:

- Les CSS Grids sont complémentaires à Flexbox et permettent de créer facilement des mises en page plus élaborées que Flexbox, sans forcément avoir des éléments de la même taille.
- Pour déclarer une grid, on déclare simplement `display: grid;` sur le conteneur : notre navigateur comprend tout de suite que nos éléments sont dans la grid.
- On définit les colonnes avec `grid-template-columns` (nombre de valeurs (largeur de chaque cadre) \Rightarrow nbres colonnes et les rangées) avec `grid-template-rows` : en fonction du nombre de valeurs passées, de nouvelles colonnes et rangées sont créées.
- En plus des unités classiques `px`, `em`, `rem` et `%`, les `fr` sont encore plus simples, et permettent d'indiquer une fraction de la grille.
- `gap` permet d'espacer les éléments entre eux.
- Les grids créent implicitement des lignes horizontales et verticales délimitant les différentes rangées et colonnes.
- Chaque élément peut avoir :
 - son propre point de départ horizontal avec `grid-row-start` ;
 - son point d'arrivée horizontal avec `grid-row-end` ;
 - son point de départ vertical avec `grid-column-start` ;
 - et son point d'arrivée vertical avec `grid-column-end` .

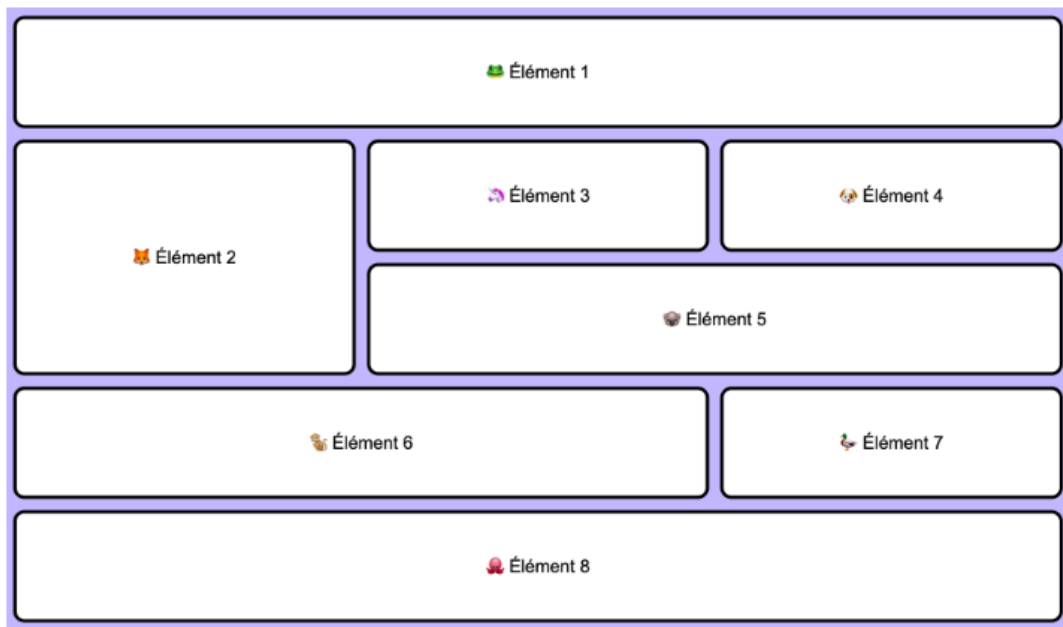
```
<div class="conteneur">
<div class="box une">🐼 Élément 1</div>
<div class="box deux">🦊 Élément 2</div>
<div class="box trois">🦄 Élément 3</div>
<div class="box quatre">🐼 Élément 4</div>
<div class="box cinq">🐼 Élément 5</div>
<div class="box six">🐼 Élément 6</div>
<div class="box sept">🐼 Élément 7</div>
<div class="box huit">🦊 Élément 8</div>
<div class="box neuf">🦄 Élément 9</div>
</div>
```

Et en CSS, on a :

```
.conteneur {
display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 100px 100px 100px 100px 100px;
```

```
    grid-gap: 10px;
}
```

Pour avoir ce genre de résultat:



on devra faire ceci :

```
<div class="conteneur">
  <div class="box une">🦊 Élément 1</div>
  <div class="box deux">🐱 Élément 2</div>
  <div class="box trois">🦄 Élément 3</div>
  <div class="box quatre">🐶 Élément 4</div>
  <div class="box cinq">🐼 Élément 5</div>
  <div class="box six">🐵 Élément 6</div>
  <div class="box sept">🦆 Élément 7</div>
  <div class="box huit">🦋 Élément 8</div>
  <div class="box neuf">🦋 Élément 9</div>
</div>
.conteneur {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 100px 100px 100px 100px 100px;
}
```



```

    grid-gap: 10px;
}
/*****pour mesurer la colonn*****/
//donc avec la boxe 1 on par exemple:
//grid-column start pour lui dire qu'on commence à la prem
//et on termine à la troisieme(4 car elle finit à la trois
.une {
    grid-column-start: 1;
    grid-column-end: 4;
}
ou sur une seul ligne:
.une {
    grid-column: 1 / 4;
}

/****pour mesurer à quelle ligne elle se trouve****/
.deux {
    grid-row-start: 2;
    grid-row-end: 4;
}
//ou
.deux {
    grid-row: 2/4
}
//elle commence à la deuxieme rangée et se termine à la 4

```

- La propriété `display` permet de changer le comportement de base d'un élément :
 - transformer un élément `inline` en `block` ;
 - inversement, un élément `block` en `inline` ;
 - mais aussi de faire un mélange des deux avec `inline-block` .
- Le positionnement relatif permet de décaler un bloc par rapport à sa position normale.


- Le positionnement absolu permet de placer un élément où l'on souhaite sur la page, au pixel près.
- Un élément A positionné en absolu à l'intérieur d'un autre élément B (lui-même positionné en absolu, fixe ou relatif) se positionne par rapport à l'élément B, et non par rapport au coin en haut à gauche de la page.
- Lorsque plusieurs éléments s'empilent, il est possible de les ordonner avec `z-index`.

en fait on fait toujours position: relative ou absolue ou sticky ou fixed et après on fait selon ça un z-index pour définir qui sera au dessus de qui.

TABLEAUX:

- Un tableau s'insère avec la balise HTML `<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 HTML avec la propriété CSS `border`. Pour coller les bordures entre elles, on utilise la propriété CSS `border-collapse`. (valeurs: collapse, separate)
- Un tableau peut être divisé en trois sections grâce aux balises HTML `<thead>` (en-tête), `<tbody>` (corps) et `<tfoot>` (bas du tableau). L'utilisation de ces balises n'est pas obligatoire.
- En HTML, on peut fusionner des cellules horizontalement avec l'attribut `colspan`, ou verticalement avec `rowspan`. Il faut indiquer combien

de cellules doivent être fusionnées.

Type de champ	Code
un e-mail	<code><input type="email"></code>
une URL	<code><input type="url"></code> On peut demander à saisir une adresse absolue, commençant généralement par <code>http://</code>
un numéro de téléphone	<code><input type="tel"></code>
un nombre entier	<code><input type="number"></code> Le champ s'affiche en général avec des petites flèches pour changer la valeur.
un curseur (aussi appelé "slider")	<code><input type="range"></code> On utilise range pour demander au visiteur une valeur comprise entre deux bornes : Audio : 

une date

Différents types de champs de sélection de date existent, comme

`<input type="date">` pour sélectionner une date :

HTML	CSS
1	<code><label for="start">Date :</label></code>
2	
3	<code><input type="date" id="start" name="trip-start"</code>
4	<code>value="2023-01-01"</code>
5	<code>min="2023-01-01" max="2025-12-31"></code>

Date :
01/01/2023
janvier 2023
L M M J V S D
26 27 28 29 30 31 1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31 1 2 3 4 5

Un champ de date et le code associé

Mais il existe des variantes :

- `<input type="time">` pour l'heure ;
- `<input type="week">` pour la semaine ;
- `<input type="month">` pour le mois ;
- `<input type="datetime">` pour la date et l'heure (avec gestion du décalage horaire) ;
- `<input type="datetime-local">` pour la date et l'heure (sans gestion du décalage horaire).



Vérifiez bien quels navigateurs gèrent ce type de champ. il n'est pas encore complètement reconnu.

une
recherche

`<input type="search">`

HTML	CSS
1	<code><label for="site-search">Rechercher :</label></code>
2	<code><input type="search" id="site-search" name="q"></code>
3	
4	<code><button>GO !</button></code>

Rechercher :

Un champ de recherche et le code associé

Pour les types de champs nombre, date et curseur, vous pouvez personnaliser le fonctionnement du champ avec les attributs suivants :

- `min` : valeur minimale autorisée ;
- `max` : valeur maximale autorisée ;
- `step` : c'est un "pas" de déplacement. Si vous indiquez un pas de 2, le champ n'acceptera que des valeurs de 2 en 2 (par exemple 0, 2, 4, 6...).

exemple:

```
<form method="get" action="">
<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</label><br>
<input type="checkbox" name="epinards" id="epinards"> <label
for="epinards">Épinards</label><br>
<input type="checkbox" name="huitres" id="huitres"> <label
for="huitres">Huitres</label>
</p>
</form>
```

Vous pouvez faire en sorte qu'une case soit cochée par défaut, avec l'attribut `checked` : `<input type="checkbox" name="choix" checked>` .

ut simplement pour que le navigateur sache de quel groupe votre sélection fait partie. Essayez d'enlever les attributs `name` , vous verrez qu'il devient possible de sélectionner tous les éléments d'options. Or, ce n'est pas ce que l'on veut ; c'est pour cela qu'on les lie entre eux en leur donnant un nom identique.

Vous noterez que cette fois on a choisi un `id` différent de `name` .

En effet, les valeurs de

`name` étant identiques, on n'aurait pas pu différencier les `id` .

En fait le name peut être le même mais l'id est toujours différent :

- Un formulaire est englobé par la balise HTML `<form></form>` à laquelle on ajoute les attributs `method` et `action`.
- On utilise ensuite d'autres balises HTML pour permettre au visiteur du site de saisir des informations :
 - la balise orpheline `<input>` pour un champ de saisie monoligne ;
 - la balise en paire `<textarea> </textarea>` pour un champ de saisie multiligne ;
 - la balise en paire `<select> </select>` suivi d'options avec la balise en paire `<option> </option>` pour une liste déroulante.
- Le champ `<input>` peut également être configuré pour saisir d'autres types de données : e-mail, URL, numéro de téléphone, date, etc.
- Un label peut être relié à n'importe quel input avec l'attribut `for` correspondant à l'attribut id pour le champ utilisé.
- Il est possible de regrouper les champs au sein d'une balise `<fieldset>` qui comporte une légende, avec la balise `<legend>`.
- On peut sélectionner automatiquement un champ en précisant l'attribut `autofocus`.
- Il est également possible de rendre un champ obligatoire avec l'attribut `required`.
- Le bouton qui permet de valider le formulaire est créé à partir d'une balise input en faisant : `<input type="submit" value="Envoyer">`

Donc un exemple est :

```
<form method="get" action="">
  <fieldset>
    <legend>Vos coordonnées</legend> <!-- Titre du fields
```

```

<label for="nom">Quel est votre nom ?</label>
<input type="text" name="nom" id="nom">
<label for="prenom">Quel est votre prénom ?</label>
<input type="text" name="prenom" id="prenom">
<label for="email">Quel est votre e-mail ?</label>
<input type="email" name="email" id="email">
</fieldset>
<fieldset>
  <legend>Votre souhait</legend> <!-- Titre du fieldset
  <p>
    Faites un souhait que vous voudriez voir exaucé :
    <input type="radio" name="souhait" value="riche" i
    <input type="radio" name="souhait" value="celebre"
    <input type="radio" name="souhait" value="intellig
    <input type="radio" name="souhait" value="autre" i
  </p>
  <p>
    <label for="precisions">Si "Autre", veuillez préci
    <textarea name="precisions" id="precisions" cols="
  </p>
</fieldset>
</form>

```

On dispose de pseudo-classes en CSS pour changer le style :
des éléments requis avec

```
:required ;
```

autofocus: pour envoyer la souris directement à cet endroit.
et des éléments invalides avec

```
:invalid .
```

N'oubliez pas non plus que vous disposez de la pseudo-classe

:focus pour changer l'apparence d'un champ lorsque le curseur se trouve à l'intérieur.

exemple: `<input type="text" name="prenom" id="prenom" autofocus>`

Exemple pour colorer le fond des champs requis :

```

:required {
  background-color:#F2A3BB;

```

```
}
```

1. `type="submit"` : **c'est celui que vous utiliserez le plus souvent**. Le visiteur sera conduit à la page indiquée dans l'attribut `action` du formulaire.
2. `type="reset"` : remise à zéro du formulaire (testez pour voir).
3. `type="image"` : équivalent du bouton `submit`, présenté cette fois sous forme d'image. Rajoutez l'attribut `src` pour indiquer l'URL de l'image.
4. `type="button"` : bouton générique, qui n'aura (par défaut) aucun effet. En général, ce bouton est géré en JavaScript pour exécuter des actions sur la page, mais nous n'apprendrons pas à le faire dans le cadre de ce cours.

L'attribut flex permet de dire qu'on prend un certain espace disponible :
exemple

flex: 1 //prend tout l'espace dispo

flex: 2; //prend 2 fois l'espace dispo.

Responsive:

syntaxe est un peu particulière, mais rassurez-vous si vous n'arrivez pas à vous en souvenir : vous pourrez toujours aller la chercher sur internet, et la copier-coller.

Voilà à quoi cela ressemble :

```
@media screen and (max-width: 1200px) {  
  /* Insérez vos propriétés CSS ici, avec vos sélecteurs*/  
}
```


Les tailles que nous indiquons avec `min-width` ou `max-width` sont souvent appelées "points de ruptures", ou *breakpoints* en anglais, car ce sont les points où le design change.

Attention, afin que les Media Queries soient prises en compte correctement sur tous les dispositifs, il est également essentiel de rajouter dans l'en-tête (partie `<head>` de notre site) la ligne suivante :

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Ici on demande à ce que le "viewport" du navigateur, c'est à dire la surface d'affichage du navigateur s'adapte à la largeur d'affichage de l'appareil. Cela permet notamment un affichage correct sur nos téléphones mobiles.

En effet les téléphones actuels ont une résolution native proche des écrans HD. Par exemple un iPhone 12Pro a une résolution native de 1 170 × 2 532 pixels. Si nous ne mettons pas la balise meta viewport, l'affichage de notre site serait le même que sur un écran de cette taille, donc on y verrait pas grand chose. Par contre avec cette balise l'affichage se fait sur une base de 390 × 844 pixels. Cela permet donc d'assurer un affichage standard sur tous les mobiles.

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

Si vous voulez que le texte ne dépasse pas les limites du paragraphe, il va falloir utiliser la propriété CSS `overflow`. Voici les valeurs qu'elle peut accepter :

- `visible` (par défaut) : si le texte dépasse les limites de taille, il reste visible et sort volontairement du bloc ;
- `hidden` : si le texte dépasse les limites, il sera tout simplement caché. On ne pourra pas voir tout le texte ;
- `scroll` : là encore, le texte sera caché s'il dépasse les limites. Sauf que cette fois, le navigateur mettra en place des barres de défilement pour qu'on puisse lire l'ensemble du texte. C'est un peu comme un cadre à l'intérieur de la page ;
- `auto` : c'est le mode "pilote automatique". En gros, c'est le navigateur qui décide de mettre ou non des barres de défilement (il n'en mettra que si c'est nécessaire). C'est la valeur que je conseille d'utiliser le plus souvent.

Avec `overflow: hidden;` le texte est donc coupé (on ne peut pas voir la suite) :

Pour adapter un site internet aux différentes tailles d'écran, plusieurs techniques sont nécessaires :

- Il faut tout d'abord réussir à sélectionner les différentes tailles d'écran disponibles, afin d'appliquer le style souhaité.
- Pour cela, on a recours aux media queries qui sont la manière de "conditionner" le style appliqué : "Si l'écran de l'utilisateur est plus petit que la taille `xxxpx` , alors appliquer ce style".
- Il est ensuite possible d'utiliser les différentes astuces :
 - `display: none` pour cacher un élément qui ne rentre pas sur les tailles d'écran plus petites ;
 - `overflow` pour permettre de scroller avec la souris dans un élément container ;
 - `min-width` et `max-width` pour définir des tailles minimum et maximum.

Bravo à vous ! Vous avez quasiment terminé ce cours !