RECAPITULATION DU CSS
LES BASIQUES

SOMMAIRE

I. LE CSS……P2

II. LES CLASSES……P4

1. Changeons d'aspect……P5

2. Les cartons de déménagements……P6

A. Les grands cartons……P7

B. Les bordures……P7

C. Les petits cartons……P9

D. Margin et Padding……P9

E. Flexbox……P11

3. Les identifiants……P12

4. La sélection mix……P13

CSS = Cascade Style Sheet

Rappel : afin de lier le fichier HTML au fichier CSS, il faut utiliser cette balise suivante entre les balises **<head></head>**

Cette balise là →

```
<link rel="stylesheet" href="nom_du_fichier.css">
```

→ Si le fichier CSS se trouve dans le même dossier que la page HTML, seul le nom du fichier suffit, si jamais le fichier CSS est dans un autre dossier, il faut indiquer le chemin précis d'où se trouve le CSS pour que l'HTML (et la machine par conséquent) puisse le retrouver. Il en va de même pour les images, les vidéos…

Rappel 2 : dans une page web, l'HTML s'occupe du fond (tout le contenu de la page), et le CSS s'occupe de la forme (l'esthétique et la beauté)

Rappel 3 : NE METTEZ PAS DE CSS DANS L'HTML (gardez tout le CSS dans son fichier respectif, on ne mélange pas…)

I. LE CSS

Grosso modo c'est comme ça →

```
le_truc_que_tu_veux_modifier{
    /*bonsoir je suis un commentaire qui te dit qu'ici
    tu peux mettre toutes les petites formules magiques pour
    rendre tout beau ton site web uwu*/
}
```

Sans le commentaire :

```
untruc{

}
```

(untruc est simplement pour indiquer qu'il faut mettre quelque chose)

→ pour cibler ce que tu veux changer dans ton HTML, tu mets le nom de la balise, la classe ou l'identifiant, et entre les accolades, toutes les choses que tu souhaites modifier, ajouter, changer, tout ce qui te plaît pour t'amuser avec ton site. En d'autres mots, c'est là que ça va être marrant parce que tu vas pouvoir jouer avec les propriétés pour changer des trucs et t'amuser !

Prenons un exemple :

Voici un paragraphe dans le code HTML (reconnaissable avec ses balises p) →

```
<p>Je suis un pauvre paragraphe tout triste :(</p>
```

Il ressemble à ça sur le site actuellement →

Je suis un pauvre paragraphe tout triste :(

Nous voulons maintenant lui mettre une petite couleur pour le rendre plus joyeux, dans ce cas, en CSS, on fait un truc comme ça →

```
p{
    color: blueviolet;
}
```

Et sur le site, cela donne ça →

Je suis un pauvre paragraphe tout triste :(

→ On met un **p** pour signifier qu'on cible **toutes** les balises **p**, puis on lui indique dans les accolades qu'on veut lui changer **sa couleur**, et on indique quelle couleur on souhaite mettre. On ferme la propriété avec un **;**

!! → Faites attention à bien respecter la syntaxe, mettre les propriétés entre les accolades et les terminer par un ;

PS : le carré qui affiche la couleur est un aperçu de base qui s'ajoute dans Visual Studio Code, donc si vous utilisez un autre éditeur de texte qui ne l'affiche pas, ne paniquez pas, vous n'avez rien fait de mal.

→ Si jamais on souhaite une couleur plus spécifique, mais que l'on ne possède pas son nom, on peut utiliser directement son code RGB qui va très bien fonctionner.

```
p{
    color: #6983FF;
}
```

→ Je suis un pauvre paragraphe tout triste :(

→ Pour l'exemple, j'ai utilisé un paragraphe, mais on peut très bien utiliser toutes les balises que l'on veut à la place. Par contre, il faut savoir que ce que j'ai effectué pour ce petit paragraphe, va également s'effectuer pour **TOUS** les autres paragraphes, le **p seul ciblant tous les paragraphes du fichier HTML.**

C'est ici qu'entrent en jeu les classes. Parce que c'est bien de colorier tous les paragraphes d'une couleur, mais si je préfère que les autres paragraphes soient en rouge, comment je fais ?

II. LES CLASSES

→ Dans un premier temps : les classes c'est génial. Les classes, cela permet de donner une sorte de post-it à plusieurs balises pour leur dire : 'vous, vous faites la même chose, et pas toi'.

Par exemple, reprenons nos paragraphes.

Voici 3 gentils paragraphes :

```
<p>Bonsoir, j'aime les pizzas</p>
<p>Bonjour, j'aime les nouilles</p>
<p>Bon après-midi, j'aime les donuts</p>
```

Bonsoir, j'aime les pizzas

Bonjour, j'aime les nouilles

Bon après-midi, j'aime les donuts

Si en CSS, on met ça :

```
p{
    color: red;
}
```

Ils deviennent comme ça →

Bonsoir, j'aime les pizzas

Bonjour, j'aime les nouilles

Bon après-midi, j'aime les donuts

Ils prennent tous la même couleur puisque en CSS on a indiqué que toutes les balises **p** seraient rouges.
Maintenant, imaginons que nous voulons que le premier paragraphe seul soit rouge, et les autres bleus.

→ C'est ici qu'interviennent les classes ! En HTML, on va donc utiliser une classe comme ça : **class= 'nom_de_classe'** **!- dans la balise ouvrante.**

Comme ça →

```
<p class="jesuisrouge">Bonsoir, j'aime les pizzas</p>
```

→ En CSS, une classe se présente comme ça avec un point devant le nom de la class

```
.jesuisrouge{
    color: red;
}
```

Au final nous avons ça en HTML →

```
<p class="jesuisrouge">Bonsoir, j'aime les pizzas</p>
<p>Bonjour, j'aime les nouilles</p>
<p>Bon après-midi, j'aime les donuts</p>
```

Et en CSS →

```
p{
    color: blue;
}

.jesuisrouge{
    color: red;
}
```

Sur le site web →

Bonsoir, j'aime les pizzas

Bonjour, j'aime les nouilles

Bon après-midi, j'aime les donuts

→ On peut mettre n'importe quoi pour la classe, mais privilégiez quelque chose d'assez logique pour pouvoir s'y retrouver plus facilement, et par exemple ne pas nommer les classes « tata » « toto ». Mais plutôt leur donner des noms en rapport avec la fonction que vous voulez leur donner.

→ La même classe peut s'attribuer à ce que l'on veut. Par exemple, je peux faire ça :

```
<p class="jesuisrouge">Bonsoir, j'aime les pizzas</p>
<p>Bonjour, j'aime les nouilles</p>
<p>Bon après-midi, j'aime les donuts</p>

<h2 class="jesuisrouge">Coucou les amis</h2>

<p>Je suis un hérisson</p>
<p>Je suis une feuille</p>
```

Bonsoir, j'aime les pizzas

Bonjour, j'aime les nouilles

Bon après-midi, j'aime les donuts

Coucou les amis

Je suis un hérisson

Je suis une feuille

→ La classe agira de la même façon sur toutes les balise où elle est assignée. Peu importe si cette balise est un **p** ou un **h**, ou une **liste**… Bref vous avez compris (j'espère).

1. Changeons d'aspect

Quelques autres balises bien pratiques :

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

Bon c'est petit, mais vous avez compris, j'espère, qu'ici, j'ai voulu mettre un fond de couleur à toute la page, ici à la balise body, qui représente toute le corps de la page.

Bien sûr, on peut aussi mettre une image de fond :

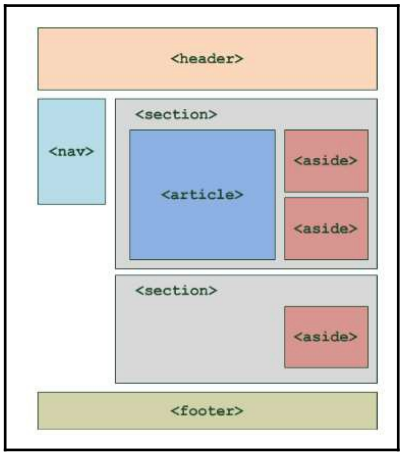
```css
body{
    background-image: url(images/David.png);
}
```

→ On peut affecter plein de truc aussi à une propriété, par exemple, si on veut répéter le fond ou ne pas le répéter, le garder fixe par rapport au reste, bref, on peut faire plein de trucs.

2. Les cartons de déménagement

→ C'est ici que les classes deviennent relativement intéressantes. Reprenons l'exemple du déménagement et des cartons.
Les balises comme ** ** (qu'on évite d'utiliser trop fréquemment !), **<section></section>**…

Ces balises là souvenez-vous →



→ Eh biens ces balises, imaginez que ce sont des cartons, et les classes c'est ce que vous marquez dessus (ce carton va dans le salon, un autre dans ta chambre…) Imaginez ces balises comme des cartons.
Prenons cet exemple :

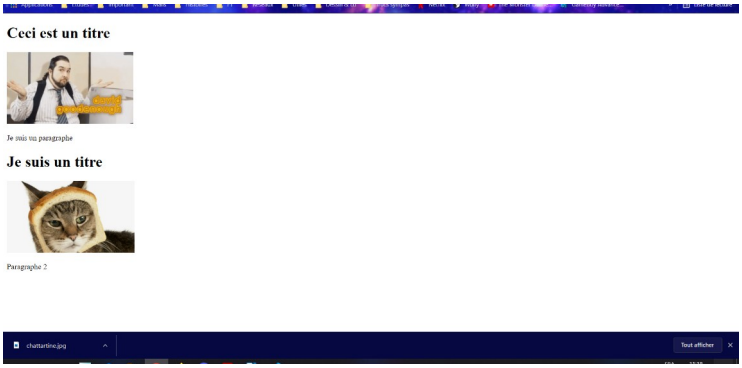
Pour le moment nous avons ça :

```html
<body>

    <h1>Ceci est un titre</h1>
    <img src="images/David.png" alt="David Goodenough">
    <p>Je suis un paragraphe</p>

    <h1>Je suis un titre</h1>
    <img src="images/chattartine.jpg" alt="Chat Tartine">
    <p>Paragraphe 2</p>

</body>
```



!! → Les images étaient trop grandes à l'origine, alors en CSS, j'ai mis la propriété qui suit pour changer leur hauteur

```
img{
    height: 150px;
}
```

→ Toutes les images auront une **hauteur** de 150 pixels. Pour modifier la largeur, on utiliser **width.**

Height → hauteur
Width → largeur

Maintenant, imaginons que je souhaite réunir, un peu comme je l'ai imagé en HTML, un grand carton avec les deux titres, les images et les paragraphes. Mais j'ai envie aussi de ranger deux petit cartons dans ce grand carton. Ces deux cartons contiennent chacun un titre, une image, et un paragraphe. Vous voyez où je veux en venir ?

A. Le Grand Carton

Bien. Faisons une **section** qu'on appellera **'grand_carton'** qui englobe tout ce qu'on a dans la balise **body** →

Voilà donc notre grand carton. Sur le site, pour l'instant ça ne changera rien visuellement.

```
<section>
    <h1>Ceci est un titre</h1>
    <img src="images/David.png" alt="David Goodenough">
    <p>Je suis un paragraphe</p>

    <h1>Je suis un titre</h1>
    <img src="images/chattartine.jpg" alt="Chat Tartine">
    <p>Paragraphe 2</p>
</section>
```

Maintenant, pour lui donner son nom de **'grand_carton'**, on met une classe dans la balise **section ouvrante**. Comme ça →

```
<section class="grand_carton">
```

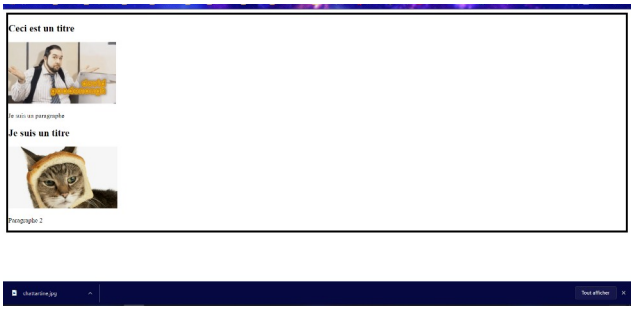
Maintenant qu'on a attribué une classe à la balise section, c'est comme si on avait noté **'grand_carton'** au marqueur sur notre carton.

B. La Bordure

Ensuite, on veut faire une bordure qui englobera tout ce qu'il y a dans **'grand_carton'**. Pour cela en CSS on effectue ça →

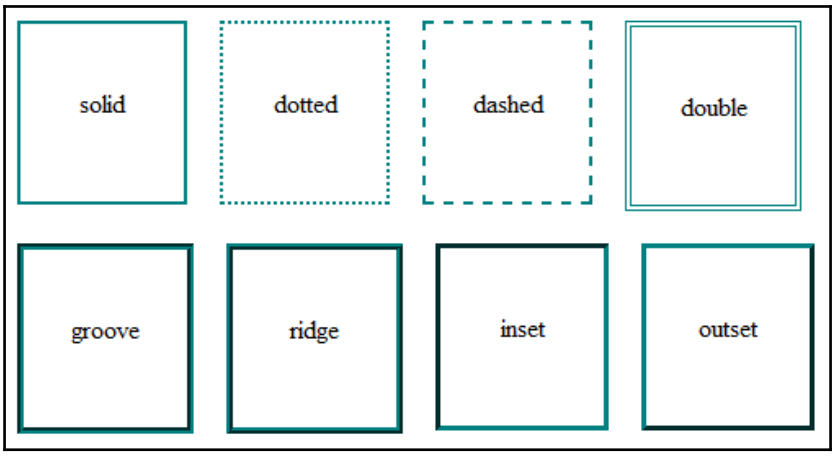
```
.grand_carton{
    border: 5px black solid;
}
```

Voilà maintenant on a notre bordure →



→ Dans le CSS, la propriété **border** possède ces valeurs actuelles :
5px = épaisseur de la bordure
black = sa couleur
solid = le type de bordure

→ il existe d'autres types de bordures pour s'amuser.



→ Pour la bordure, nous ne sommes pas obligés d'avoir les quatre cotés.

border-top *= bordure du dessus*
border-bottom ***=*** *bordure du dessous*
bordure-left *= bordure gauche*
bordure-right *= bordure droite*

→ Vous vous doutez bien que si vous souhaitez la bordure complète (les 4 côtés) mettez simplement **border**. Automatiquement cela mettra les 4 côtés de la bordure.

C. Les Petits Cartons

→ Pour obtenir nos deux petits cartons, on va changer et utiliser des balises **<article></article>** et leur assigner chacun la classe **‘petit_carton’**.

En HTML ça donne donc ça →

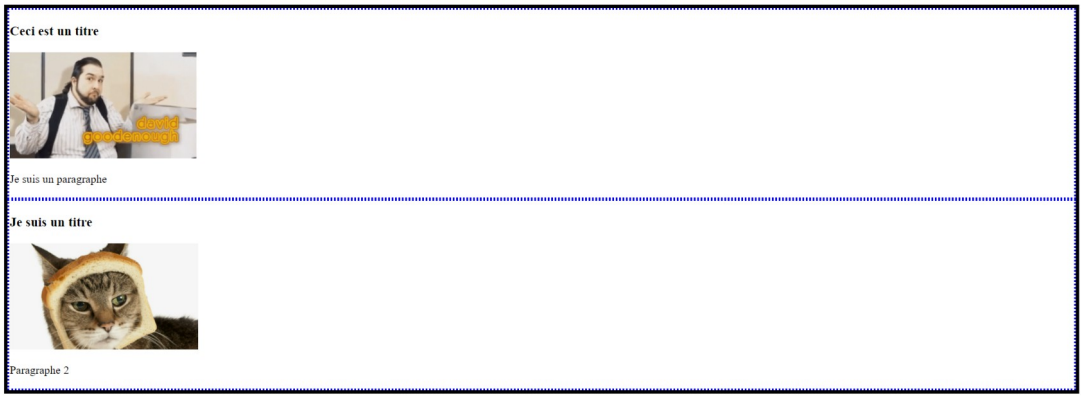
```html
<section class="grand_carton">
    <article class="petit_carton">
        <h1>Ceci est un titre</h1>
        <img src="images/David.png" alt="David Goodenough">
        <p>Je suis un paragraphe</p>
    </article>

    <article class="petit_carton">
        <h1>Je suis un titre</h1>
        <img src="images/chattartine.jpg" alt="Chat Tartine">
        <p>Paragraphe 2</p>
    </article>
</section>
```

On va s’amuser à mettre une bordure à nos deux cartons :

Comme vous le voyez, j’ai changé les valeurs de la bordure pour que ce ne soit pas trop morne →

```css
.petit_carton{
    border: 3px blue dotted;
}
```



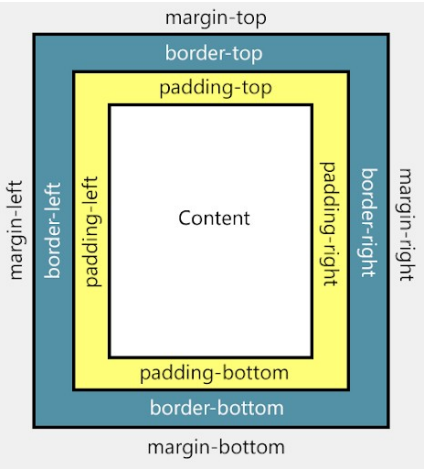
Comme vous le voyez, on peut distinguer désormais qu’il existe 2 petites boites, mais ce n’est pas très beau, les bordures sont collées…

D. Margin et Padding

En CSS, il existe margin et padding. Ces deux propriétés fonctionnent comme des marges.

Margin = marge extérieur (avec le bord de la page)
padding = marge intérieure (avec la bordure de l’élément)

Pour imager, ça donne ça →



! → Comme pour les bordures, on peut régler seulement un des côtés si l'on ne veut pas faire tous les côtés.

→ Le 'Content' c'est le contenu. Ici notre contenu c'est ce qu'il y a dans les cartons qu'on a fait.

Maintenant, appliquons un **margin** à nos petits cartons.

En CSS on fait donc ça →

```css
.petit_carton{
    border: 3px blue dotted;
    margin: 20px;
}
```

Maintenant c'est un peu plus clair !
→

Mais maintenant on aimerait bien mettre nos petits cartons côte à côte.
On va donc utiliser **display**.

Et pour les mettre en ligne, on utilisera ça :
Regardez bien on a rajouté **display** →

```css
.petit_carton{
    border: 3px blue dotted;
    margin: 20px;
    display: inline-block;
}
```

C'est déjà mieux maintenant !
→

→ Si maintenant je vous montre le **padding**, cela donnerait quelque chose comme ça :

```
.petit_carton{
    border: 3px blue dotted;
    margin: 20px;
    display: inline-block;
    padding: 10px;
}
```

SANS PADDING **AVEC PADDING**

→Regardez bien il y a une différence !

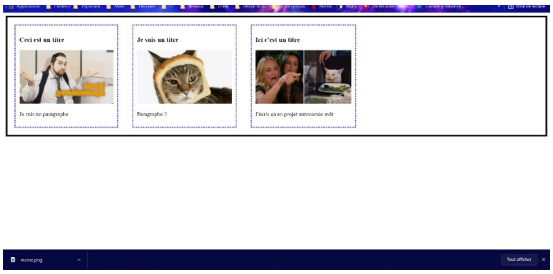
!!! → Globalement il y a vraiment plein de façon de s'amuser avec le CSS, dans le cours de la prof il y a quelques autres propriétés, sinon vous pouvez me demander ou chercher sur internet

E. Flexbox

→ Pour les mises en page je vous ai montré vite fait **inline-block**, mais en vérité, utilisez **flexbox**. Flexbox, c'est une nouvelle façon de mise en page beaucoup plus pratique pour les mises en page.

Flexbox c'est le principe des cartons, jusque là rien de change. Mais dans le CSS, on doit impérativement mettre un **display : flex** pour indiquer que toute la boite va être flexible. Ensuite, on va voir comment aligner nos cartons (en colonne, en ligne…)

```
.grand_carton{
    border: 5px black solid;
    display: flex;
    flex-direction: row;
}
```



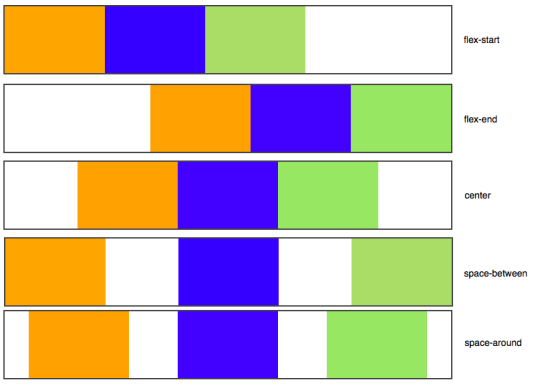
```
.grand_carton{
    border: 5px black solid;
    display: flex;
    flex-direction: column;
}
```

ps : l'autre image est en dessous, il faut juste scroll

Après que vous ayez rajouté un **display:flex**, c'est là que ça devient plus amusant.

Flex-direction = indique la direction que prend les boites
row = en ligne
column = en colonne
row-reverse = en ligne mais les éléments sont placés dans l'autre sens (david goodenough est en dernier tandis que le meme de la dame qui crie sur le chat est en premier)
column-reverse = colonne inversée

Justify-content = emplacement des éléments (comme l'image là) →



flex-wrap = si l'élément dépasse il retournera à la ligne, par exemple vous pouvez essayer de zoomer sur une page, elles retourneront à la ligne, n'ayant plus de place pour se mettre côte à côte.

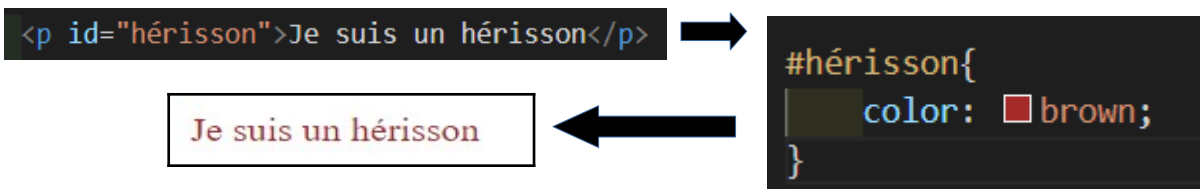
Align-items = aligner les éléments sur l'axe secondaire, à l'horizontal.

Voilà quelques fonctionnalités géniales que flexbox nous permet de faire qui sont vraiment pratiques pour bouger nos chers cartons.

3. Les identifiants

→ Vous vous souvenez de **id='nom'** ? Avec le liens dans la page, l'exemple pour naviguer dans la page Wikipédia sans scroller 300ans… ?

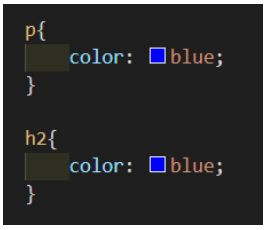
→ L'id fonctionne globalement comme les classes, mais contrairement aux classes, l'id ne peut être assignée qu'à un seul élément, tandis que les classes peuvent être assignées sur plusieurs balises à la fois dans l'HTML.



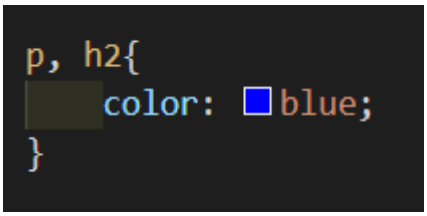
Bon en gros, les id c'est juste pour un seul truc, alors que les classes tu peux les mettre à gogo sur tout ce que tu veux, mais ça fonctionne pareil dans l'ensemble.

4. La sélection mix

→ pour éviter de se prendre le chou, parfois, on peut mettre plusieurs balises pour les mêmes propriétés.
Si par exemple, on veut que tous les paragraphes et les titres h2 soient en bleu, au lieu de faire deux catégories en CSS de cette manière :



On peut simplement faire ça :



C'est plus simple et lisible, et on gagne de la place !
→ Bien sûr il est est possible de le faire avec plein d'autres balises, j'ai seulement prise celles-ci comme exemple.

→En CSS, il y a beaucoup de choses que l'on peut faire, ici ne sont renseignés que certaines bases, mais d'autres aides vont suivre celle-ci. Comme dans l'autre doc, n'hésitez pas à regarder sur internet, et surtout comprendre ce que vous faites, c'est impératif.

Hasphodelle49