Les exemples et les bouts de codes de ce document peuvent être essayés sur le site codepen.io. pour avoir un aperçu immédiat du résultat et mieux en comprendre le comportement.

Principe de flux HTML

Le document HTLM répond à une logique de flux, ce flux fait que les éléments sont physiquement dépendants les uns des autres. Si la logique standard de flux des éléments n'est pas changé les dimensions d'un élément, par exemple, vont avoir une influence sur les éléments qui l'entourent et les repousser.

Exemple : Créez un document, modifier les propriétés de dimensions d'un objet et constater les effets sur les éléments adjacents.

Les unités

Il est possible en CSS d'utiliser un large panel d'unités. Des unités fixes et des unités relatives et permettent de modifier la dimension d'un élément, de changer le corps de la police utilisée, de changer sa couleur…

Unités de dimensions

On découpe généralement ces unités en deux groupes, les unités *fixes* et les unités *variables*.

Les unités fixes

Ce sont des **unités invariables**, elles ne **s'altèrent pas** selon le contexte dans lequel elles sont utilisées, seule **la résolution du terminal chargé de faire le rendu va faire varier l'aspect** de ces unités.

- *px*, le pixel. Un pixel est une unité fixe représentant un point sur un

écran. Dans l'absolu elle est invariable mais est dépendante de la seule résolution du terminal affichant ce pixel;

- *cm* et *mm*, le centimètre et le millimètre comme sur vos règles;
- *in* un *inch* correspond à 2,54 cm;
- *pt*, le point typographique;
- *pc* le pica, unité typographique. 1pc = 12pt;

Les unités variables

Ce sont des valeurs dont les dimensions ne sont pas absolues, elles varient selon un contexte (la dimension de la fenêtre du navigateur, la taille de police du document principal…) et sont en général dépendantes d'éléments parents ou de *container*.

Ces unités sont %, *em* et *ex*.

- %, le fameux pourcentage. Un élément à qui ont donne une dimension en pourcentage aura la dimension calculée en contexte de son élément parent.Si un élément de 200 pixels contient un élément de 50% de largeur, cet objet aura une dimension calculée de 100 pixels;
- *em*, cette unité est issue du monde de la typographie et réprésente la largeur d'un caractère typographique, le M capital. Elle est surtout utilisée pour les polices de caractères et permet de gérer un affichage harmonieux du texte en fixant des relations proportionnelles entre les éléments. Une taille de corps indiquée en *em* se fera toujours par rapport à la taille de la police de l'élément parent. Considérons un élément dont la taille de police est fixée à 2em, si son parent a une taille de police indiquée à 20 pixels, cet élément aura une taille de police de 2 x 20px = 40px;
- *ex* est équivalent à *em* mais prend en référence la hauteur d'un x

minuscule comme référence (la hauteur des bas-de-casse);

Les unités de couleur

Les couleurs sur les écrans se notent en trois valeurs de base, **RVB**, Rouge, Vert et Bleu. Ces trois couleurs sont les composantes de base de et **leur intensité va faire varier la luminosité et la valeur de la couleur**.

L'intensité de chaque couleur s'indique dans une valeur numérique qui va de 0 à 255 pour chaque composante, rouge, vert et bleu.

Les couleurs héxadecimales

La notation dite *héxadécimale*, encore très utilisée en CSS indique les valeurs de ces composantes RVB par le biais de 3 paires de chiffres et lettres et utilisent une base 16 incluant les caractères suivants : 0123456789abcdef.

La couleur rouge qui en notation RVB classique est *255, 0, 0* sera *ffff00* en héxadécimale.

En CSS cette couleur s'utilise en la précédant d'un dièse.

```
/* Mes textes en rouge */
.red {
    color: #ffff00;
}
```

Couleurs RGB, RGBa, HSL et HSLa

La notations RGB revient à la notation classique des couleurs sur une base allant de 0 à 255 pour chacune des composante. 0 étant noir et 255 le maximum de lumière présent dans la couleur.

```
/*  Exemple de couleur RGB, ici du rouge */
.rouge {
    color: rgb(255, 255, 0);
}
```

Le RGBa ajoute une composante appelée *alpha*, en CSS cette composante permet de manipuler 10 niveaux de transparence (de 0 à 1) pour la couleur indiquée.

```
/*  Exemple de couleur RGBa, ici du rouge à 50 %
transparent */
.rouge {
    color: rgba(255, 255, 0, 0.5);
}
```

Les notation HSL et HSLa sont moins communes mais manipulent les composantes *Hue* (la teinte de la couleur), *Saturation* et *Light* la luminosité de la couleur.

Les couleurs nommées

Il est aussi possible de nommer les couleurs utilisées. Une gamme de 256 couleurs nommées existe et peut aussi être utilisée.

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

Photoshop comme la plupart des outils graphiques disposent d'outils permettant de récupérer ces couleurs sous ces différents type de notation qui sont des standards (sauf les couleurs nommées).

Un article de Wikipedia très complet sur les couleurs utilisées sur le

web.

Les URLs

Une *URL* ou Unique Ressource Locator permet d'importer une ressource à un document. Cet *URL* est une adresse web permettant d'accéder à la ressource souhaitée.

Il existe deux manières de relier des ressources entre-elles, de manière *absolue* ou *relative*. Une *URL absolue* ne prend pas en compte la position de la page ou de la ressource appelante comme référence contrairement à l'*URL relative* qui est relative à la ressource appelante.

xoxoxoxoxo **Exemple** xoxoxoxoxoxo

La valeur en CSS comme en HTML ne change pas selon le type de relation.

En HTML

```
<!-- Lien absolu -->
<a href="http://www.apple.com">Le site d'Apple</a>

<!-- Lien relatif -->
<a href="../contact.htm">Contactez-nous</a>
```

En CSS

```
/* Appel absolu */
.element {
    background: red
url(http://www.apple.com/monimage.jpg);
}

/* Appel relatif */
.element {
    background: red url(../img/monimage.jpg);
}
```

Les CSS, *Cascading StyleSheets*

Ou *styles en cascade*, ce n'est pas un langage de programmation mais un **langage descriptif** conçu pour définir le style des éléments HTML, leur **propriété visuelles et parfois même orales**.

Pour cela les CSS ont **3 composantes principales** :

- **le sélecteur** indiquant le ou les objets que l'on souhaite modifier;
- **la propriété** que l'on souhaite modifier;
- **la valeur** de cette propriété.

Une déclaration CSS

```
    /* Commentaire */
    selecteur {
        propriété : valeur;
    }
```

L'ensemble des **déclarations** est se que l'on nomme le **bloc de déclaration**, et ce bloc de déclaration va appliquer l'ensemble de ses **propriétés** à **un ou plusieurs éléments** présents dans la

sélection.

Les commentaires ne sont que pour les auteurs, autrement dit le navigateur ne les prends pas en compte dans le rendu.

Les sélecteurs

CSS donne la possibilité de sélectionner un ou des éléments par le biais d'une syntaxe assez simple. Cette sélection peut comporter un ou plusieurs éléments de la page.

Le sélecteur universel

Ce sélecteur permet de sélectionner l'intégralité des éléments présents dans la page ou dans un élément.

```
* { propriété : valeur; }
```

Ici tous les éléments sont sélectionnés. Il n'est pas recommandé d'utiliser ce sélecteur car il se révèle assez *lourd* à gérer pour le navigateur. Il est très gourmand en ressource.

La sélection par balise

La sélection par balise ou *tag*, elle permet de sélectionner un groupe d'éléments utilisant ce *tag*.

Exemple de sélection par balise

Dans le document HTML

```
<p>Bonjour je suis le paragraphe</p>
```

Et dans le fichier CSS

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

Ici le ou les `p` présents dans la page seront de couleur bleue.

La sélection par l'attribut *class*

Comme nous l'avons vu précédemment il est possible d'ajouter un attribut à un élément HTML, l'attribut *class* est particulier car il permet de faire appartenir des **éléments à un groupe d'objets** que l'on peut modifier soit par des CSS pour appliquer un style, soit par du Javascript pour en modifier le comportement.

En CSS un sélection par *class* commence par un « . » suivi du nom de la *class* des éléments que l'on souhaite modifier.

Exemple de sélection par *class*

Extrait de mon document HTML

```
<h1 class="couleur">Ici le titre de mon document</h1>
<p class="couleur">Un paragraphe de texte…</p>
```

et le bloc CSS me permettant de sélectionner et de modifier l'apparence de ces éléments

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

Dans l'exemple précédent le navigateur va chercher l'intégralité des éléments dont l'attribut *class* a la valeur *couleur* et va leur appliquer la propriété de couleur avec la valeur *blue*. Ces deux éléments auront donc la couleur bleue.

La sélection par l'attribut *id*

Comme pour l'attribut *class* chaque élément de la page HTML peut avoir un attribut *id*. Cet attribut est en revanche particulier car **il est unique par document HTML**, il ne peut y avoir plusieurs éléments portant la même valeur d'*id*.

L'*id* est comparable à une *identité*, ce qui est unique à chacun d'entre nous.

En CSS un sélection par *id* commence par un « # » suivi du nom de l'*id* de l'élément que l'on souhaite modifier.

Exemple de sélection par *id*

Dans mon document HTML

```
<h1 id="titrePrincipal">Ici le contenu de mon élément</h1>
```

Et dans mon fichier CSS

```
#titrePrincipal {
    color: red;
    font-size: 60px;
    }
```

La navigateur va chercher dans le document HTML l'élément portant l'attribut *id* ayant la valeur *titrePrincipal* et lui appliquer les propriétés indiquées.

La sélection multiple ou sélection par groupe

Comme nous l'avons vu précédemment avec la sélection par l'attribut *class* il est possible de sélectionner plusieurs éléments au sein d'un document HTML et de leur appliquer des propriétés communes.

Mais afin d'éviter d'avoir à ajouter des *class* à tous les éléments et conserver un document HTML le plus léger possible, il est aussi possible d'appliquer des propriétés communes à plusieurs éléments par un seul bloc de déclaration en regroupant les sélecteurs précédemment vus.

Afin d'effectuer une sélection multiple il faut cumuler dans la sélection les différents sélecteur que l'on souhaite par une simple virgule « , ». Le groupe de sélecteur peut être hétérogène et comporter des sélection par tag, par attribut, par id…

Exemple de sélection multiple

Extrait du document HTML

```
<h1 id="grosTitre">Un titre de niveau 1</h1>
<ul>
    <li>Élément 1 de la liste</li>
    <li>Élément 2 de la liste</li>
</ul>
<h2 class="titreSection">Titre de niveau 2</h2>
<h2>Un autre titre de niveau 2</h2>
```

Extrait du document CSS

```
ul, .titreSection, #grosTitre {
    color: brown;
}
```

Dans cet exemple, la sélection s'applique aux trois objets indiqués (`ul, .titreSection et #grosTitre`) qui sont alors de couleur marron.

La sélection hiérarchique

La sélection hiérarchique est dans la nature même des CSS, c'est de la que vient le nom de *cascade*.

Ce mode de sélection permet d'appliquer des styles à un élément en conséquence de sa présence ou non dans un certain contexte. Cet élément est-il un enfant de cet élément ?

Pour effectuer une sélection hiérarchique, il suffit de séparer les niveaux hierarchiques de sélection par un simple espace « » et dans l'ordre suivant :

```
grand-parent parent enfant { propriété: valeur;  }
grand-parent enfant { propriété: valeur;  }
```

Exemple de sélection hiérarchique

Extrait du document HTML

```
<h1>Un cours de <em>CSS</em></h1>
<p>Bonjour tous le monde, nous faisons des <em>CSS</em>
ici !</p>
<ul>
    <li>Les sélecteurs par <em>class</em></li>
    <li>Les sélecteurs par <em>id</em></li>
</ul>
```

Dans mon fichier CSS

```
h1 em {
    color: blue;
}
p em {
    color: yellow;
}
```

Dans cet exemple la sélection s'effectue sur les *em* présents dans la page, mais les contextes sont différents dans chacune des déclarations. La première déclaration, `h1 em`, ne va sélectionner que les élément `em` lorsqu'ils sont dans un élément `h1` et la seconde pour les mêmes éléments lorsqu'ils sont présents dans un `p`.

Sélecteurs d’attributs

Ce sélecteur fait partie des sélecteurs avancés de CSS. Sa syntaxe peut être complexe, mais c’est un outil extrêmement puissant de sélection. La sélection par attribut permet de sélectionner un élément par le nom de son attribut mais aussi sur la valeur de cette attribut.

Exemples de sélection

Extrait HTML

```
<ul>
    <li><a href="http://www.apple.com">Apple</a></li>
    <li><a
href="http://www.microsoft.com">Microsoft</a></li>
    <li><a href="addres.php">Adresse</a></li>
</ul>
```

Et le fichier CSS

```
a {color: blue;}
a[href="http://www.microsoft.com"] {color: yellow;}
a[href*=php] {color: green;}
```

Dans cet exemple, les 3 liens ont des couleurs différentes. L’ensemble des liens a une couleur bleue `a {color: blue;}`, mais certaines règles qui viennent après spécifient des couleurs différentes aux liens avec leurs attributs propres. `a[href="http://www.microsoft.com"] {color: yellow;}` vient sélectionner le lien dont l’attribut `href` a pour valeur exacte : *http://www.microsoft.com*. La dernière règle recherche tous les liens dont l’attribut `href` contient `php`, cela concerne donc le dernier lien de l’exemple mais cela pourrait concerner tous les liens contenant `php` dans leur attribut `href`.

Infos détaillées sur la sélection par attribut.

Sélection par descendance directe

Ce type de sélection vient compléter la sélection hierarchique mais est stricte sur le niveau de descendance. On sélectionne un élément seulement si il est un enfant direct de son parent.

Cette sélection se sasait en séparant les éléments de la hiérarchie par un « > ».

Exemple de sélection par descendance directe

Dans le document HTML

```
<p>Ce <strong><em>texte est passionant</em></strong>.
</p>
<p>Ceci est un autre <em>paragraphe</em>.</p>
```

En CSS

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

Dans cet exemple, seul le deuxième `em` sera en bleu car il est un enfant direct. L'élément `em` présent dans la première ligne de code HTML est un petit-enfant et n'est donc pas sélectionné par le navigateur.

Sélection adjacente

Nous pouvons sélectionner des éléments **seulement si ils sont précédé d'un certains autre élément**, on parle de sélection adjacente.

Exemple de sélection adjacente

```
<h2>Titre de la page</h2>
<p>Texte d'introduction, lorem ipsum…</p>
<p>Encore du texte lorem upsum…</p>
```

et en CSS

```
h2 + p {font-weight: bold}
```

Dans l'exemple précédent je souhaite ques tous les paragraphes (`p`) précédés d'un `h2` soient affichés en gras, pour présenter un texte d'intro à chacune de mes pages. Le premier `p` sera donc en gras, le second ne verra pas ses propriétés modifiées.

Les propriétés

Maintenant que vous savez sélectionner des éléments, il faut en faire quelques chose et c'est l'objet des propriétés et de leurs valeurs.

Il est possible en CSS de modifier l'intégralité des aspects d'un élément. (À noter que les CSS s'applique surtout au visuel d'un élément mais permet aussi de gérer la restitution auditive des éléments).

L'objet de ce document n'est pas de détailler l'intégralité des propriétés existantes mais d'aborder les propriétés les plus utilisées et d'expliquer la plupart des unités qui y sont rattachées.

Color

Propriété de modifier la couleur du texte.

Exemple

Dans mon document HTML

```
<div class="coul">Texte qui doit être bleu</div>
```

Dans mon fichier CSS

```
.coul {color: blue}
```

Il est à noter que les couleurs s'indiquent de plusieurs manières en CSS.

Background, le fond des éléments

Très utilisée, c'est la propriété qui permet de manipuler le fond des éléments. Elle permet d'en modifier les aspects suivants :

- La couleur du fond;
- La présence d'une image de fond et de fournir l'adresse de l'image qui sera utiilsée;
- Le mode de répetition (pour un motif par exemple), la position de cette image ainsi que son mode d'accrochage au *scroll* de la page.
- Les versions récentes de CSS permettent l'ajout de multiples images mais le support des navigateurs n'est pas encore très largesur ce sujet.

Exemple complet de propriété de fond

```
.element {
    background: red url(monimage.jpg) scroll no-repeat
bottom left;
}
```

Cet exemple montre la propriété *raccourcie*, elle regroupe tous les aspects du fond qui sont modifiables aussi via des propriétés particulières dans une seule déclaration.

Dans l'exemple les éléments ayant la classe *.element* auront :

- un fond rouge;
- Une image *monimage.jpg*;
- Cette image ne sera pas répétée;
- Cette image sera positionnée dans le bloc en bas à droite;
- L'image suivra le *scroll* de l'utilisateur.

Et cet exemple pourrait être découpé de la manière suivante avec toutes les propriétés de fond détaillées de la manière suivante:

```
.element {
    /* Le fond est de couleur rouge */
    background-color: red;

    /* Il a une image */
    background-image: url(monimage.jpg);

    /* L'image n'est pas en motif */
    background-repeat: no-repeat;

     /* L'image est positionnée en bas à gauche */
    background-position: bottom left;

    /* L'image suit le déplacement de l'ascenceur */
    background-attachment : scroll;
}
```

Mais cette notation détaillée est lourde et ne s'utilise pas directement de cette manière. Elle sert en général à dissocier d'un groupe un objet ayant une particularité visuelle en manipulant une des propriété du fond de cet élément.

Font

```
p {
    font:  italic bold small-caps 18px/1em Helvetica,
sans-serif;
}
```

Cette propriété permet de contrôler tous les aspects de la police de caractère, le corps, l'interligne, les variantes (*small-caps*), la graisse et la famille utilisée.

Comme pour la propriété background, la propriété *font* est une propriété *raccourcie*. Mais elle est assez peu utilisée, car a une contrainte quand à l'ordre des sous-propriétés.

L'exemple pourrait se décliner par des sous-propriétés de la manière suivante :

```
p {
    /* Ce texte est affiché en gras */
    font-weight: bold;

    /* La variante est en petites capitales */
    font-variant: small-caps;

    /* Corps de la police */
    font-size: 18px;

    /* Interligne */
    line-height: 1em;

    /* Texte en italic, je modifie le style */
    font-style: italic;

    /* Je modifie la famille de police utilisée */
    font-family: Helvetica, sans-serif;

}
```

font-weight, modifie la graisse de la police, les valeurs utilisables sont *bold* ou *normal*. La valeur *normal* permet de rétablir l'aspect par défaut de la police.

font-variant, crée une variante de la police,les valeurs sont *small-caps* et *normal*.

font-size modifie le corps de la police. Les valeurs peuvent utiliser plusieurs unités (pixel : px, point : pt, em, ex, cm…). Nous verrons plus tard ces unités et leur valeur réeeles.

font-style change le style de la police, les valeurs sont *italic*, *oblique* (une fausse italique) et *normal*.

font-family change la police utilisée pour un élément. La valeur va indiquer une série de familles de polices correspondant aux préférences de l'auteur. Certaines familles de polices n'étant pas présentes sur toutes les plateformes il est possible d'indiquer de manière progressive les dégradations possibles sur le choix des polices à utiliser. La première valeur indiquée correspond à la fonte *idéale* puis la seconde à un *second choix* ainsi de suite pour arriver sur les familles *génériques* disponibles sur toutes les plateformes.

```
.element {
    font-family: mabellepolice, mamoinsbellepolice,…,
famille générique;
}
```

Ces familles génériques sont les suivantes :

```
* *sans-serif*, une police à patins (Helvetica, Geneva,
Arial…);
* *serif*, une police à patin (Times, Georgia…);
* *cursive*, une police cursive proche des écritures
manuelles (Zapfino…);
* *fantasy*, une police décorative (Papyrus…);
* *monospace*, une police à espacement régulier
(Monaco, Courrier…).
```

L'ajout de cette famille est obligatoire dans la propriété *font-family* ou *font*.

line-height

La propriété *line-height* permet de gérer l'interligne de l'élément selon la valeur indiquée.

```
.element {
    line-height: 20px;
}
```

À lire à propos de la hauteur de ligne et des unités.

width, *height*

Chaque élément peut avoir une hauteur et une largeur grâce à ces propriétés.

```
.element {
    width: 100%;
    height: 50px;
}
```

L'élément de l'exemple aura une largeur de 100% de son élément parent et une hauteur fixe de 50 pixels.

margin, *padding*

border

display

Position

Documents à consulter tous les jours avant d'aller dormir

- Les bases de CSS
- Une histoire des CSS
- Liste complète de propriétés CSS
- Présentation intéractive de quelques schémas de sélection

- La référence CSS de Mozilla et plus particulièrement les références à propos de la sélection.
- Connaître la disponibilité de certaines propriétés et sélecteurs de CSS selon le navigateur
- Ensemble de bonnes pratiques pour bien documenter ses documents CSS : https://github.com/necolas/idiomatic-css/tree/master/translations/fr-FR