***Le css***

Informations générales :

Dans le meta de l’HTML, on va lier le fichier HTML à un fichier CSS avec la meta

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

Directement dans le head.

Quand on code en CSS, dans une balise, on met entre crochets {

}

Par exemple : tag{

Color :red ;

Size :16px ;

}

On finit toujours les différentes modifications par un ;

Exemple de page style.html :

html{

margin:0;

padding:0;

background-image:url(galaxie.jpg);

}

body{

font-size:100%;

font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background-color:black;

width: 960px;

padding:0;

margin:0 auto;

border-radius:20px;

}

h1{

color:red;

text-align:center;

font-family:'Times New Roman', Times, serif;

}

p{

color:white;

text-align:center;

background-color:brown;

margin:0 200px;

border-radius:15px; }



Quelques astuces :

Il faut toujours penser à vérifier ce que l’on fait avec F12 sur les navigateurs pour voir les tailles de nos différentes balises. Cette commande permet de voir quelle balise à un problème de taille par exemple.

Quelques balises :

Pour arrondir les angles d’un border : (dans la balise que l’on veut modifier)

border-radius:20px;

Pour intégrer une image en fond : (dans le html :{ )

background-image:url(galaxie.jpg);

Pour positionner le texte : (left,right,center,justify)

text-align:center;

Pour modifier la couleur du texte : (par exemple dans les p{

color:red;

Pour modifier la couleur du fond :

background-color:black;

Pour modifier la police d’écriture :

font-family:'Times New Roman', Times, serif;

Pour mettre un texte en gras :

font-weight:bold;

Pour souligner un texte : (ici souligné en pointillés)

text-decoration:underline dotted

Mettre son texte en italique :

font-style: italic;

Mettre une div à côté d’une autre au lieu qu’elle aille en dessous :

float: left; Left ou right selon le côté où l’on veut placer la div.

Pour que tous les padding et margin de la page soient à 0 ; de base :

\*,html{

margin:0; ce n’est pas encore officiellement la norme selon W3C.

padding:0;

Pour déplacer une div en vertical dans sa div :

vertical-align: center ; Ou bottom, top,…

Pour gérer ce qui dépasse d’une div :

overflow:hidden; hidden, visible, auto (pour avoir un scroll vertical) scroll (pour un scroll vertical et horizontal).

Pour gérer juste la dernière div enfant dans un ul par exemple :

.myclass:last-child{ First-child pour la premiere div,

border-bottom:0px;

}

Pour modifier une div enfant qui n’est ni la première ni la dernière :

.myclass:nth-child(4){ le numéro de la div à modifier entre parenthèses.

border-bottom:0px;

}

Les classes :

Elles commencent toujours par un point comme par exemple :

.p1{

color:yellow;

background-color: blue;

font-style: italic;

}

Une classe sert à définir un style particulier pour chaque chose qui sera tag sous le tag du paragraphe. Par exemple, dans l’HTML

<span class="p1">statuuntur Epigonus</span>

dira que ce paragraphe est tag comme étant classé sous la classe .p1 et donc ce qui a été défini dans le CSS s’appliquera à l’HTML.

Si on veut que les paragraphes pairs soient en noir et les impairs en rouge, il suffit de créer deux classes et à chaque fois appliquer la classe voulue au paragraphe quand on le crée.

Il est possible de mettre plusieurs classes pour une seule balise.

Les ID

Contrairement à la classe, l’ID créé dans le fichier CSS ne peut être appliqué qu’une seule fois.

Il commence par un #

#entete{

font-size: xx-large;

color:red;

text-decoration: underline

}

Positionner les éléments :

Le padding : c’est l’espace par rapport au bord (qu’il soit invisible ou pas).

Le margin est la distance entre le bord et le prochain objet.

Border :   
-Solid  
-Dash  
- …

padding

Zone de texte,  
Image,  
…

Margin

Quand on met par exemple :

Margin : 0px 200px ;

La première valeur équivaut à la distance haut-bas avec l’objet et la seconde la distance gauche-droite.

Si on met quatre valeurs, elles signifieront la distance avec les côtés dans le sens des aiguilles d’une montre.

On peut mettre par exemple :

Margin :0px auto ;

Le auto permet de centrer directement peu importe la taille de l’écran.

Le padding est l’espace entre un objet et ses bordures. Par exemple, un <p> à une bordure invisible qui entoure le texte. Le padding définit l’espace entre le texte et la bordure.

Le margin définit la distance entre deux objets. Par exemple, pour notre zone de texte, le margin représentera l’espace entre les bordures invisibles de la zone de texte et celles de la page web.

Il faut toujours regrouper par classes les choses qui peuvent l’être pour pouvoir tout modifier en une fois :

html{

margin: 0;

padding: 0;

}

.div1{

border-right:solid green 10px;

margin-top:50px;

margin-left:60px;

}

.div2{

border-right:solid blue 10px;

margin-top:130px;

}

.div3{

border-right:solid orange 10px;

margin-top:210px;

}

.aspect{

border-left:solid black 1px;

border-bottom:solid black 1px;

border-top: solid red 10px;

background-color:silver;

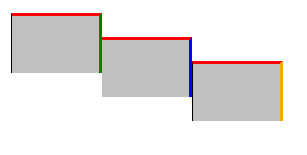
width:290px;

height:190px;

float:left;

}

Ici, il y a trois div différentes qui ont pas mal de choses en commun. Chaque div à sa classe pour les choses qui lui sont propres et une quatrième classe a été créée pour regrouper leurs points communs. Ainsi, si on veut par exemple décider que le background passe de silver à black, en allant modifier la classe .aspect on touche directement à toutes les divs. Pratique quand on doit modifier en une fois un grand nombre de divs. Le résultat :



Les sélecteurs :

Un sélecteur est dans un fichier CSS ce qui sera avant le {qui va permettre de modifier des éléments HTML : ça peut être un tag HTML, comme un paragraphe par exemple ou une image, mais aussi une classe ou un ID.

.div2{

**Il est possible de mettre plusieurs sélecteurs d’affilé (séparés par une virgule) si l’on veut qu’ils fassent la même chose :**

div,p,h5 {

color:blue;

text-align:justify;

}

**Si on met un espace entre les sélecteurs, c’est le deuxième sélecteur qui sera modifié uniquement s’il descendant du premier :** ici, les paragraphes hors div ne seront pas affectés.

div p{

text-align:justify;

color:black;

}

**On peut également utiliser le symbole > si l’on veut modifier uniquement l’enfant direct :**

p>img {

border:solid black 1px;

}

Ici, l’image dans le paragraphe sera modifiée (enfant direct), mais celle qui est dans le tableau qui est lui-même dans un paragraphe non, car c’est un enfant indirect :

<p> <img src="exemple.jpeg"> </p> sera modifiée.

<p> <table> <img src="exemple.jpeg"> </table> </p> Ne le sera pas.

Les positions :

Il existe quatre types de positions :

**-statique :** la position de base, inutile de le spécifier. Le bloc ne bougera pas.

**-relative :** permet de déplacer un objet mais sa place restera quand même. Pour une image par exemple, il y aura un trou où elle était initialement et elle sera déplacée par-dessus le contenu, donc elle peut recouvrir quelque chose. En relative left et right sont inversés.

**-fixed :** le fixed permet de fixer par rapport au viewport (le navigateur). Un bloc fixed sera toujours au même endroit peu importe la taille de l’écran. Par exemple, un footer fixed en bottom. Contrairement au relative, le bloc (une image par exemple) est retirée du flux pour être replacée quelque part donc sa place initiale est perdue. C’est très utile si on veut qu’un élément ne bouge pas même si on scroll. Par exemple, en fixant le header et le footer, on pourra scroll tout en gardant le menu en haut de page et le bas restera en bas.

**-absolute :** les objets fixés en absolute resteront fixés par rapport à leur contenant. Par exemple, une image fixée en absolute dans une div en son centre restera toujours en son centre, même si on scroll. Dans un article, une image positionnée en absolute restera toujours par exemple au-dessus du texte même si on scroll.

Pour placer une div en absolute par rapport à une div parent, il faut ***ABSOLUMENT*** que la div parent soit placée en autre chose que static. En static la div a une position « de base » et n’est pas considérée comme positionnée d’une manière particulière. La div fille en absolute ne se placera donc pas par rapport à elle, mais par rapport à celle encore parent à son parent, donc en général le HTML. Il faut donc mettre le parent en position relative sans donner de coordonnées pour qu’elle ne se déplace pas, et ainsi la positionner. Dans ce cas, la div fille en absolute se placera bel et bien par rapport à son parent, à l’intérieur de celui-ci..

**Subseed (left,right,bottom,top):**après la position, on précise la direction et la distance de déplacement :

position:relative;   
left:150px;  
bottom:150px;

***la position est TOUJOURS indiquée avant le subseed.***

**z-index :** le z-index permet d’organiser les couches. Plus le z-index est élevé, plus l’objet sera positionné au-dessus des autres. Pour placer quelque chose en dernière couche, par exemple une image en fond d’un texte, on mettra une valeur négative.

.class{

position:absolute;

bottom:10px;

z-index:-1;

}

Le box sizing :

De base, le navigateur va calculer la taille d’une div sur son contenu, et il faudra par la suite calculer de nous-même l’ajout de la border, du padding et du margin pour avoir un résultat propre.

IE a proposé l’idée que la div pouvait calculer avec les borders pour plus de clarté, et après débats au sein de la communauté des devs, il a été admis que la solution d’IE n’était pas mauvaise.

Il est possible de changer la méthode de calcul avec la commande :

box-sizing: border-box;

pour calculer version IE ou

box-sizing: content-box;

}

Pour la version classique.

C’est très intéressant par exemple pour contenir les borders d’une div qui fait 100% de largeur afin que la border ne sorte pas du navigateur. C’est aussi mieux pour placer les divs les unes à côté des autres pour éviter les décalages à cause des borders de 1px.

Pour les choses qui ne sont pas encore reconnues par la W3C (encore en draft mais très utile et déjà développées par certains navigateurs) il est possible d’adapter par rapport à chaque navigateur en mettant une balise particulière :

-moz-box-align: 50px; Pour Mozilla

-ms-box-align:50px; Pour Microsoft

-webkit-box-align: 50px; Pour Chrome et Safari

-o-box-align: 50px; Pour Opéra

box-align:50px; La solution définitive est toujours en dernier.

Comme ça quand la W3C aura validé et que les navigateurs seront à jour, il n’y aura pas de conflits, et en attendant, comme la balise est avant la balise définitive le HTML la prendra en compte en premier.

Le flex :

Le display :flex ; permet de placer des div enfants côte à côte à l’intérieur d’une div parent.

#parent{

padding: 0;

margin: 0;

width:100%;

display:flex;

flex-wrap: wrap;

justify-content: center;

.

Pour placer une div en flex, il y a 9 positions :

***Justify-content*** : flex-start / center / flex-end. Dans l’axe des x (horizontal).

***Align-items :*** flex-start / center / flex-end. Dans l’axe des y (vertical).

start start center end-start

Start center center center end center

Start end center end end end

Il existe d’autres propriétés :

***Justify-content :***

***Space-between :*** permet de faire un espace équivalent entre chaque div concernées par le flex, mais elles sont collées aux bords de la div.

***Space-around :*** permet de faire un espace équivalent, mais en plus les divs sur les bords auront un espace équivalent avec le bord de la div.

display:flex;

justify-content: space-around;

flex-flow:row wrap;

***On peut définir une direction aux div qui seront soumises au flex :***

***Flex-direction :***

***Row*** dira que les flex sont en lignes

***Row-reverse*** les mettra en lignes inversées

***Column*** les mettra en colonnes

***Column-reverse*** les mettra en colonnes inversées :

display:flex;

flex-direction:column-reverse;

***Flex :auto*** dans la div à déplacer rendra sa taille adaptable pour occuper tout l’espace restant. Ça peut être utile par exemple quand on développe un nav et qu’on veut que les li occupent tout l’espace de celui-ci.

Le ***flex-wrap :wrap :*** adaptera les div à la taille de la div parent pour ne pas qu’elles sortent de celle-ci.

La balise ***flex-flow*** : combine les balises ***flex-direction*** et ***flex-wrap***:

flex-flow: row-reverse wrap; Permet de gagner en lignes de code.

Dans les div éléments, on peut les organiser dans l’ordre qu’on veut en mettant order :chiffre dans leur classe :

.div1{

order:3; La div sera donc la numéro 3 de la liste.

}

On peut déplacer (en hauteur seulement) une div par rapport aux autres avec la balise :

align-self: flex-start; ou center ou flex-end selon le positionnement voulu.

Donner un effet actif à une image ou une div :

transition:1s;

transform:rotate(360deg) scale(1.6);

Dans la div que l’on veut voir bouger :

La transition va déterminer en combien de secondes l’action se fera et le transform contiendra l’effet souhaité et à l’échelle voulue (par exemple une rotation de 360 degrès).

li:hover{

color:red;

text-shadow: 10px 6px 2px purple;

font-size:1.1em ;

text-decoration:underline;

font-style:italic;

transition:1s;

transform:rotate(360deg) scale(1.6);

font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

Les ::before et ::after :

.titlecolor::before{

content:" ";

position: absolute;

background: purple;

transition: 1s;

height: 3px;

display:block;

width: 0;

}

.titlecolor:hover::before{

width: 100%;

}

**::Before** mettra ce qu’on veut toucher avant le contenu ciblé dans la balise (ici .titlecolor)

**::after** le mettra après.

**Le content : « » ;** est obligatoire dès qu’on fait before ou after. Si on veut mettre du texte avant le content, ou mettra le texte dans les « ».

Ici, le but est de faire une barre qui avance au survol de la souris et qui repart de l’autre sens quand la souris repart. La position absolute est utile car comme la barre sort du flux, elle ne déplace pas le menu vers le bas au survol si la largeur de la barre est assez élevée.

Le background indique la couleur de la ligne.

Le display :block est important car il va définir que la barre est au-dessus ou en-dessous du contenu (ici au-dessus car ::before) .

**Le .titlecolor :hover ::before** dit que dès qu’on survole, la largeur passe de 0% à 100% de la div, et la transition du dessus indique que ça se fera en 1s. :hover est obligatoirement avant le ::before.

Pour faire partir la ligne de droite à gauche, il faut ajouter un **right :0 ;**