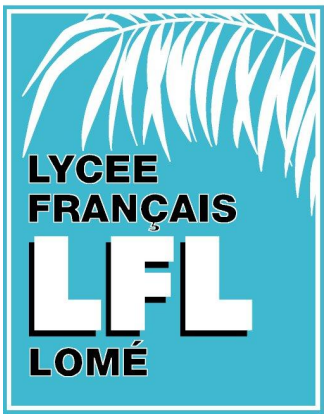
Les feuilles

de styles CSS



Sommaire :

I. Présentation :

Les feuilles de styles en cascades *(Cascading StyleSheets - CSS)* font actuellement partie intégrante de la conception de pages Webs et constituent l'outil le plus important pour la mise en page des pages Webs.

Les feuilles de styles sont le complément obligatoire du HTML pour le design des sites Web. De plus, la possibilité de placer les définitions dans un fichier externe permet à l'ensemble des pages d'un site d'utiliser les mêmes définitions de styles *(pour les titres, les commentaires...)* : cela permet d'obtenir un formatage homogène pour l'ensemble d'un site.

Elles ont été mises en place en 1996 et officialisées par le W3C *(World Wide Web Consortium à l'origine de la plupart des standardisation du Web)*. Le W3C a introduit la version CSS2 en mai 1998. Mais, il faudra attendre aux alentours de 2003 pour voir les navigateurs prendre en compte ce standard.

Comme dans tous les normes, pour le Web, quelques différences d'interprêtation existent entre les différents navigateurs.

II. Les propriétés de styles :

Ce paragraphe présente la totalité, ou presque, des propriétés de styles accessibles dans le langage CSS.

Certaines de propriétés agissent sur les bords des éléments HTML ; ces propriétés prennent de 1 à 4 valeurs. Ainsi, lorsqu'une valeur est passée en argument, elle sera affectée aux quatres bords de l'élément.

Le tableau ci-dessous, indique pour ces propriétés, l'affectation de chaque argument en fonction du nombre d'argument fournis à la propriété de style.

	Bordure affectée			
Exemple	Haut	Gauche	Bas	Droit
prop=*a*	a	a	a	a
prop=*a b*	a	b	a	b
prop=*a b c*	a	b	c	b
prop=*a b c d*	a	b	c	d

A. Formatage de police :

font-family

Valeur par défaut : `dépend du navigateur` Héritage : `oui`

Cette propriété permet de sélectionner la police de caractères utilisée par l'élément courant.

Les valeurs de la propriété :

- une chaîne de caractères contenant le nom d'une police.
  Pour palier à la possibilité d'abscence d'une police sur le poste du client, il est possible de préciser plusieurs noms de polices séparés par une virgule ; la première police disponible chez le client est utilisée.
- Voici des noms faisant appel à des polices génériques :
  serif ; *sans-serif* ; *cursive* ; *fantasy* ; *monospace*

font-size

Valeur par défaut : `medium` Héritage : `oui`

Cette propriété définit la taille de la fonte courante. Voici les quelques valeurs prédéfinies de cette propriété :

- Les valeurs suivantes définissent la taille de la fonte de manière absolue :
  xx-small ; *x-small* ; *small* ; *medium* ; *large* ; *x-large* ; *xx-large.*
- Les deux valeurs suivantes permettent de modifier la taille d'un élément relativement à la taille de l'élément parent *(celui qui contient leélément courant)* :
  larger ; *smaller*

Cette propriété accepte également les valeurs suivantes :

- Une longueur *(voir en annexe).* Par habitude, l'unité de mesure en point est davantage utilisée.
- Un pourcentage exprimant la taille utilisée relavement à celle de l'élément parent.

font-style

Valeur par défaut : `normal` Héritage : `oui`

Cette propriété contrôle l'inclinaison de la police. Voici les valeurs possibles :
normal ; *oblique* ; *italic.*

Dans le cas où la valeur *italic* est absente de la police courante, la valeur *oblique* est utilisée.

font-weight

Valeur par défaut : `normal` Héritage : `oui`

Cette propriété modifie la "*graisse*" de la fonte courante. Voici les valeurs prédéfinies pour cette police :

- *normal* ; *bold*
- *bolder* et *lighter* permettent de définir la graisse de la fonte relativement à celle de l'élément parent.

Il est également possible de lui attribuer un nombre représentant la graisse de la fonte parmi :

100 ; *200* ; *300* ; *400* ; *500* ; *600* ; *700* ; *800* ; *900*

En pratique, il est rare de trouver des polices comportant 9 niveaux de graisse.

font-variant

Valeur par défaut : `normal` Héritage : `oui`

Avec la valeur *small-caps*, cette propriété affiche le contenu de l'élément en petites capitales : les minuscules apparaîssent comme des petites majuscules.

Avec la valeur *normal*, la fonte courante conserve son aspect.

color

Valeur par défaut : `dépend du navigateur` Héritage : `oui`

Cette propriété définit la couleur de la fonte de l'élément courant. Elle admet pour valeur une couleur *(voir en annexe)*.

line-height

Valeur par défaut : `normal` Héritage : `oui`

Cette propriété détermine la hauteur d'une ligne. Voici les valeurs acceptées :

- *normal* : la hauteur de la ligne est définie par la police courante.
- Un nombre : ce nombre a un effet de facteur multiplicateur de la hauteur courante de la ligne.
- Une longueur : elle spécifie la taille de la hauteur. *(pensez à utiliser les unités relatives "em" et "ex" - voir en annexe).*
- Un pourcentage : ce pourcentage représente la hauteur de chaque ligne de l'élément relativement à la hauteur de ligne définie par la fonte courante.

font

Valeur par défaut : `voir chaque propriété`Héritage : `oui`

Cette propriété permet de définir d'un seul coup les valeurs des six propriétés suivantes :

font-style, ***font-variant***, ***font-weight***, ***font-size***, ***lien-height***, ***font-family***

Elle admet comme valeur les valeurs d'une ou de plusieurs des propriétés citées ci-

dessus *(la valeur de **font-size** est obligatoire).*

Les valeurs pré-définies par le système d'exploitation du client permettent de rappeler certaines fontes localement utilisées :

- *caption* représente la fonte utilisée pour les contrôles *(boutons, menu déroulants)* du système ;
- *icon* représente la fonte utilisée pour les icônes;
- *menu* représente la fonte utilisée dans les menus ;
- *message-box* représente la fonte utilisée dans les boîtes à dialogues ;
- *small-caption* représente la fonte utilisée pour les polices de petites tailles ;
- *status-bar* représente la fonte utilisée par la barre de statut de la fenêtre.

B. Contrôle du texte :

En simplifiant, il existe deux types d'éléments HTML :

- Les *éléments en bloc* : ils sont utilisés pour créer des paragraphes. Ces éléments s'étalent sur toute la largeur de la page et peut s'étendre sur plusieurs lignes.
- Les *éléments en ligne* : ces éléments sont destinés à être placé dans le flux du texte ; ils seront donc rajoutés à la ligne courante : par exemple `i`, `span`.... Ces éléments ne peuvent contenir des éléments en blocs.

Pour plus de détails, allez voir dans le chapitre "*Positionnement et affichage des éléments*" *(page 13)*.

text-indent

Valeur par défaut : `0` Héritage : `oui`

Dans un paragraphe, la première ligne peut subir un retrait pour mettre en évidence le début d'un paragraphe.

Sa valeur pourra être :

- une *longueur* (voir appendice). Une valeur négative donnera un décalage de la première ligne vers la gauche et une valeur positive vers la droite.
- ou un *pourcentage* désignant une longueur relative à la largeur de l'élément parent.

Cette propriété s'applique uniquement aux éléments de type bloc

text-align

Valeur par défaut : `left pour les occidentaux` Héritage : `oui`

Cette propriété permet d'aligner à gauche ou à droite, de centrer ou de justifier le

contenu de l'élément.

- Pour appliquer à un élément de type bloc, voici les valeurs de cette propriété :
  left ; *right* ; *center* ; *justify*
- Appliqué à l'ensemble des cellules d'une colonne d'un tableau, on peut aligner toutes ces cellules à l'aide d'un caractère *(alignement de nombre avec leur virgule)*. La valeur de la propriété sera alors une chaîne de caractère contenant ce caractère *(un exemple est donné sur le site W3C)*

Cette propriété s'applique uniquement aux éléments de type bloc ; on pourra simplifier en disant aux éléments créant un paragraphe.

vertical-align

Valeur par défaut : `baseline` Héritage : `non`

Cette méthode ne s'applique qu'aux tableaux ; elle permet de contrôler l'alignement vertical des cellules d'une même ligne : elle modifie l'emplacement de la ligne de base. Hors des tableaux, il n'existe pas de moyen automatique d'aligner verticalement des éléments : il faut passer par le positionnement absolu..
Voici les valeurs de cette propriété :

- *baseline* ; *sub* ; *super* ; *top* ; *text-top* ; *middle* ; *bottom* ; *text-bottom*
- Sa valeur pourra être une *longueur* ou un *pourcentage (relativement à la hauteur de la ligne)* : cette propriété avec une valeur positive élève la ligne de base de l'élément et pour une valeur négative, elle l'abaisse.

text-decoration

Valeur par défaut : `none` Héritage : `non`

Cette propriété permet au contenu de l'élément :

- d'être souligné : *underline*,
- d'être surligné : *overline*,
- d'être barré : *linethrough*,
- de clignoter : *blink*,
- d'annuler ce type de formatage : *none*.

letter-spacing

Valeur par défaut : `normal` Héritage : `oui`

Cette propriété modifie l'espacement entre les lettres d'un mot *(espaces compris)*. Ses valeurs peuvent être :

- soit *normal* : l'espace est définit par la police de caractère courante.

- soit une longueur *(voir en appendice)*.

word-spacing

Valeur par défaut : `normal` Héritage : `oui`

On utilise cette propriété pour modifier l'espace inter-mots. Ses valeurs sont :

- soit *normal* : l'espace entre les mots est défini par la police courante.
- soit une longueur *(voir en appendice)*.

text-transform

Valeur par défaut : `none` Héritage : `oui`

Il est possible de modifier transformer les majuscules en minuscules du contenu d'un élément et vice-versa à l'aide de cette propriété. Voici les valeurs prises par celle-ci :

- *capitalize* : transforme la première lettre de chaque mot en majuscule.
- *uppercase* : tous les caractères passeront en majuscules.
- *lowercase* : tous les caractères passeront en minuscules.
- *none* : aucun effet n'est apporté.

white-space

Valeur par défaut : `normal` Héritage : `oui`

Cette propriété permet de contrôler le comportement du navigateur face aux coupures de lignes ou à une succession d'espaces blancs. Voici les valeurs prises :

- *normal* : ceci est le mode par défaut des navigateurs. Plusieurs espaces ou sauts de ligne dans le code seront interprêtés comme un seul espace.
- *pre* : le navigateur affiche tous les blancs *(même se répétant)* et les sauts de lignes présents dans le contenu de l'élément.
- *nowrap* : les blancs successifs sont fusionnés mais les retours à la ligne du contenu de l'élément sont affichés.

C. Espace dans les boites :

Un élément HTML est généralement contitué d'une bordure *(invicible par défaut)* et de son contenu. Les deux propriétés qui sont présentées ci-dessous permettent de définir l'espacement autour et à l'intérieur de l'élément :

- L'espace laissée entre la bordure d'un élément et son contenu est appelé marge interne *(KompoZer l'appelle "espacement")*.
- L'espace laissé entre la bordure de l'élément et les autres éléments *(élément parent ou élément voisin)* s'appele la marge extérieur *(KompoZerl'appelle la "marge")*

Il est plus évident de faire la différence entre ces deux espacements lorsque l'élément possède une bordure visible ou une couleur de fond.

Ces deux propriétés peuvent prendre de 1 à 4 nombres comme arguments, séparés par des espaces. Rendez vous à la page 4, pour comprendre la signification des arguments en fonctions de leurs nombres.

margin

Valeur par défaut : `0` Héritage : `non`

Cette propriété définit la marge d'un élément : c'est à dire l'espacement entre les bords d'un élément et son extérieur. Cette dimension est exprimée par :

- une *longueur (voir en appendice)* ;
- un *pourcentage* relativement à la taille de l'élément parent *(généralement* `body` *- le document entier).*

On définit séparemment la marge de chaque côté d'un élément à l'aide des propriétés :

margin-top ; ***margin-right*** ; ***margin-bottom*** ; ***margin-left***

padding

Valeur par défaut : `0` Héritage : `non`

Cette propriété définit l'espacement d'un élément : c'est à dire l'espacement entre les bords d'un élément et son contenu. Cette dimension est exprimée par :

- une *longueur (voir en appendice)* ;
- un *pourcentage* relativement à la taille de l'élément parent *(généralement* `body` *- le document entier).*

On définit séparemment l'espacement de chaque côté d'un élément à l'aide des propriétés :

padding-top ; ***padding-left*** ; ***padding-bottom*** ; ***padding-right***

D. Bordure :

Un élément HTML possède toujours une bordure, même si celle-ci est invisible par défaut. A l'aide des styles CSS, on peut contrôler son épaisseur, sa couleur et son apparence. Ce dernier a la valeur par défaut *none* empêchant l'affichage de la bordure.

border-width

Valeur par défaut : `medium` Héritage : `non`

Cette propriété définit l'épaisseur d'une bordure. Sa valeur est à choisir parmi :

- les valeurs prédéfinies suivantes :

  thin (mince) ; *medium (moyen)* ; *thick (épais)*

- ou une *longueur* positive *(voir en appendice).*

Cette propriété prend de 1 à 4 arguments : elle affectera séparemment ou communément les quatres bordures de l'élément *(voir page 4)*.

Il est également possible d'affecter séparemment chacune des bordures de l'élément à l'aide des propriétés suivantes :

border-top-width ; ***border-left-width*** ; ***border-bottom-width*** ; ***border-right-width***

border-color

Valeur par défaut : `valeur` *color* Héritage : `non`

Cette propriété définit la couleur de la bordure :

- une couleur *(voir en appendice)*
- la valeur *transparent* rend la bordure invisible.

Cette propriété prend de 1 à 4 arguments *(voir page 4)*

On définit séparément la couleur de chacune des bordures à l'aide des propriétés :

border-top-color ; ***border-left-color*** ; ***border-bottom-color*** ; ***border-right-color***

border-style

Valeur par défaut : `none` Héritage : `non`

Cette propriété définit le style de la bordure : simple trait, double trait, effet relief . . .
Cette propriété prend les valeurs suivantes :

none ; *hidden* ; *dotted* ; *dashed* ; *solid* ; *double*

groove ; *ridge* ; *inset* ; *outset*

Cette propriété prend de 1 à 4 arguments *(voir page 4)*.

Le style d'une seule bordure est affecté par l'une des propriétés suivantes :

border-top-style ; ***border-left-style*** ; ***border-bottom-style*** ; ***border-right-style***

border

Valeur par défaut : `voir sous-propriétés` Héritage : `non`

Cette propriété combine les trois propriétés de formatage de la bordure vues précédemment ; elle prend de 1 à 3 valeurs représentant les valeurs prises par les trois propriétés vu précédemment.
Seule la valeur de ***border-style*** est obligatoire, pour annihiler la valeur par défaut *none* ; sinon votre bordure restera invisible.

Cette propriété affecte les valeurs transmises aux quatres bordures. Il existe également les propriétés permettant d'affecter séparemment chacune des bordures :

border-top ; ***border-left*** ; ***border-bottom*** ; ***border-right***

E. Couleurs et images d'arrière plan :

background-color

Valeur par défaut : `transparent` Héritage : `non`

La propriété ***background-color*** définit la couleur de l'arrière plan. Sa valeur est à choisir parmi :

- une couleur *(voir en appendice)*,
- la valeur *transparent*.

background-image

Valeur par défaut : `none` Héritage : `non`

La propriété ***background-image*** définit l'image de fond de l'élément courant. Sa valeur est :

- l'URL de l'image *(voir en appendice)*,
- la valeur *none* pour spécifier qu'aucune image n'est affichée en arrière plan de l'élément.

background-repeat

Valeur par défaut : `repeat` Héritage : `non`

On définit à l'aide de la propriété ***background-repeat*** la manière dont l'image remplit le fond de lélément. Les valeurs possibles sont :

- *repeat* : l'image est répétée pour remplir le fond de l'élément.
- *no-repeat* : l'image est affichée une seule fois.
- *repeat-x* : l'image est répété uniquement horizontalement.
- *repeat-y* : l'image est repétée uniquement verticalement.

background-position

Valeur par défaut : `0% 0%` Héritage : `non`

Cette propriété indique la position initiale de l'insertion de la première image dans l'élément. Le point d'insertion est considérée par rapport au coin supérieur gauche de l'élément et celui de l'image.

- Les deux coordonnées peuvent être exprimées en longueur ou en pourcentage. Si une seule valeur est fournie, alors elle sera utilisée pour les abscisses et les ordonnées.
- A l'aide de deux de ces valeurs prédéfinies :
  - Pour les abscisses : *left*, *center*, *right*.
  - Pour les ordonnées : *top*, *center*, *bottom*.

L'utilisation des pourcentages dans l'exemple :

background-position: *10% 30%*

signifie *(bon courage pour la compréhension)*:

- que la ligne verticale découpant l'élément à 10% de la largeur *(en partant de la gauche)* est aligné avec la ligne verticale découpant l'image à 10% *(en partant de la gauche)*,
- et il en sera de même pour une ligne horizontale déterminant le placement vertical de l'objet *(en partant du haut)*.

background-attachment

Valeur par défaut : `scroll` Héritage : `non`

En supposant que l'élément *(body par exemple mais pas obligatoirement)* possède des barres de défilement, la propriété ***background-attachment*** permet de contrôler le positionnement de l'image lorsque le client fait défiler le contenu de l'élément. Cette propriété accepte les valeurs :

- *fixed* : l'image d'arrière plan reste fixe, pendant que le contenu de l'élément défile ;
- *scroll* : l'image d'arrière plan défile de la même façon que le contenu de l'élément.

background

Valeur par défaut : `voir chaque propriété`Héritage : `non`

Cette propriété permet d'appeler, en une seule fois, es propriétés précédentes :

background-color ; ***background-image*** ; ***background-repeat*** ***background-attachment*** ; ***background-position***

Chaque argument passé à cette propriété sera transmise à la propriété correspondante.

F. Positionnement et affichage des éléments :

Normalement, un élément arrivant au navigateur du client est placé à la suite de son prédécesseur et ainsi de suite pour former des lignes. Le navigateur se chargeant alors de déterminer

l'endroit de coupure pour former les lignes ; cette façon de composer la page s'appelle le *flux courant du texte*. Une page entièrement créée de la sorte donne l'impression d'une page de texte. Pour donner un côté plus attrayant à un site, il est souvent nécessaire de placer une image, un encadré à un endroit précis de la page.

Les feuilles de style CSS permettent de sortir les éléments du flux normal du texte et de les placer à un endroit précis. On présente dans ce paragraphe, les différentes propriétés utilisées pour contrôler la position d'un élément dans la page.

Pour comprendre l'action des propriétés suivantes, il est bon de garder en mémoire que les éléments HTML peuvent être présentés en deux grands groupes :

- Les éléments HTML `div` , `p` ,... sont de type "*en bloc*". Par défaut, ils prennent l'ensemble de la largeur de la page et peuvent contenir des sauts de lignes, d'autres paragraphes...

- Les éléments `br` , `img` ne définissent pas un contenu et les éléments `span` , `i` ... s'adapteront à la taille de leur contenu. Ces derniers ne doivent pas contenir de saut de ligne, ni délé-ments de type "*en bloc*".
  Ces éléments sont dits de type "*en ligne*"

display

Valeur par défaut : `inline` Héritage : `non`

Cette propriété définit le type de l'élément. Voici les deux valeurs qui nous intéressent pour cette formation :

- *inline* pour un élément de type "*en ligne*" ;
- *block* pour un élément de type "*en bloc*".
- avec la valeur *none*, l'élément n'est pas affiché et aucune place ne lui est réservé dans le flux du texte *(à la différence de la propriété visibility)*.

Voici les nombreuses autres valeurs que peut prendre cette propriété. Internet Explorer ne les prend pas toutes en charge et elles ne seront pas discutées ici :

inline ; *block* ; *list-item* ; *run-in* ; *inline-block* ;
table ; *inline-table* ; *table-row-group* ; *table-header-group* ;
table-footer-group ; *table-row* ; *table-column-group* ;
table-column ; *table-cell* ; *table-caption*

Pour mieux appréhender la propriété suivante, il faut rappeler que les élément HTML affichés sont tous contenus dans l'élément `body`. Celui-ci représentant l'espace d'affichage offert par le navigateur. Ainsi, tous les éléments HTML affichés possédent au moins un élément parent *(l'élément qui contient l'élément courant)*.

position

Valeur par défaut : `static` Héritage : `non`

Cette propriété définit le mode de placement de l'élément :

- *static* : l'élément est placé dans le flux normal de l'élément qui le contient, tout repositionnement est impossible.
- *relative* : l'élément est placé dans le flux normal du texte. Son emplacement est réservée dans le flux courant du texte est réservé puis il est possible de décaler sa position relativement à sa position normale.
- *absolute* : l'élément est placé hors du flux normal du texte et aucun espace ne lui est réservé. Sa nouvelle position est indiquée à l'aide des propriétés ***top***, ***left*** *(mais aussi* ***bottom***, ***right****)* et mesurée relativement au premier des éléments parents ayant un positionnement *relative* ou *absolue* ou alors l'élément body.
- *fixed* : L'élément est sorti du flux normal du texte. Son positionnement est mesuré à l'aide de la fenêtre : il est insensible au défilement de la page par le client ; ne pas confondre la fenêtre d'affichage avec l'élément body.

top ; left ; bottom ; right

Valeur par défaut : `auto` Héritage : `non`

Lorsque le mode de positionnement a une valeur différente de *static*, la position de l'élément est définie à l'aide de ces quatres propriétés.

Ces propriétés définissent l'espacement entre le bord haut *(resp. gauche, bas, droit)* de l'élément et le même bord du conteneur de l'élément. Mais le conteneur en question est différent suivant le type de positionnement utilisé :

- Dans le cas d'un positionnement *relative*, le conteneur est l'espace normalement occupé par l'élément dans le flux courant du texte.
- Dans le cas d'un positionnement *absolute*, le conteneur est le premier élément-parent en positionnement non-static ; si un tel élément n'existe pas, l'élément body est utilisé.
- Dans le cas d'un positionnement *fixed*, le conteneur est la fenêtre d'affichage.

Ces quatres propriétés prennent les valeurs suivantes :

- une *longueur (voir en appendice)*. Les valeurs négatives sont autorisées.
- un *pourcentage* présentant une longeur relativement à la boite conteneur de l'élément.
- ou la valeur *auto* et laisse ainsi le navigateur positionner l'élément.

z-index

Valeur par défaut : `auto` Héritage : `non`

En positionnement absolu, plusieurs éléments peuvent se chevaucher. Normalement, le dernier élément en position absolu arrivé dans le flux courant du texte est placé au premier plan ; la propriété ***z-index*** permet de modifier ce comportement.

Cette propriété prend pour valeur des entiers. L'élément ayant la plus haute valeur pour cette propriété est placé au premier-plan.

width ; height

Valeur par défaut : `auto` Héritage : `non`

Pour des éléments de type "*en bloc*" et en positionnement *relative* ou *absolu*, ces deux méthodes permettent respectivement de fixer la largeur et la hauteur de l'élément. Leur valeur est une dimension représentée par :

- une *longueur (voir en appendice).*
- un *pourcentage* relativement à la boite conteneur de l'élément.
- la valeur *auto* laisse le navigateur choisir.

min-width ; min-height ; max-width ; max-heigh

Valeur par défaut : `0/none` Héritage : `non`

Ces propriétés permettent d'affecter à l'élément des dimensions minimales ou maximales pour un élément "*en bloc*" qui ne soit pas positionné en *static*.

Les valeurs prises par ces fonctions sont une *longueur* ou un *pourcentage (voir détails sur la propriété précédente).*

Ces propriétés sont actuellement interprêtées par FireFox mais pas encore par Internet Explorer

Voici quelques propriétés sur l'affichage des éléments :

visibility

Valeur par défaut : `visible` Héritage : `oui`

Cette propriété permet d'afficher ou de cacher un élément. Les valeurs prises par cette propriété sont :

- *visible*
- *hidden*
- *collapse*
- *inherit*

La valeur *collapse* est un synonyme de *hidden* sauf si elle est utilisée par les tableaux : elle permet d'effacer une ligne ou une colonne mais l'écartement des lignes ou la largeur du tableau ne se trouve pas affecter de cette disparition.

Les deux propriétés suivantes permettent de contrôler le comportement d'un élément dont les

dimensions sont fixés et dont le contenu dépasse les dimensions de celui-ci :

overflow

Valeur par défaut : `visible` Héritage : `non`

Lorsque les dimensions du contenu dépassent celle du conteneur, cette propriété permet de gérer l'affichage de l'élément. Voici les valeurs prises par cette propriété et leurs effets :

- *visible* : le contenu n'est pas rogné et apparaît à l'extérieur de la boîte.
- *hidden* : la partie "*en trop*" n'est pas affichée.
- *scroll* : la partie "*en trop*" est rognée mais reste accessible via des barres de défilement rajoutées à l'élément.
- *auto* : le choix de la gestion du débordement est laissé au navigateur.

Cette propriété fonctionne uniquement pour les éléments de type "*en bloc*".

clip

Valeur par défaut : `auto` Héritage : `non`

Cette propriété ne s'applique qu'aux objets en positionnement absolu. Elle permet de rogner la partie visible de l'élément. Les valeurs acceptées sont :

- *auto* : la zone de rognage est la boite entière.
- une *zone rectangulaire* définie par *rect(haut,droite,bas,gauche)* où les quatres valeurs représentent les *longueurs* séparant la zone de rognage de chacun des bords.

  rect(2px, 15px,7px, 5px) l'affichage de l'élément est rogné de 2 pixels sur le haut ; 15 pixels sur la gauche...

On dit qu'un élément est en position flottante, lorsqu'il est retiré du flux courant du texte, placé sur un des bords de la page et que le flux est déversée pour remplir le côté de la page laissée libre.

float

Valeur par défaut : `none` Héritage : `non`

Cette propriété permet à l'élément de flotter sur la gauche ou la droite dans le flux courant du texte. Sa valeur peut être :

- *left* : l'élément est posé sur le bord gauche de son conteneur et le flux courant se déverse sur la droite de l'élément flottant.
- *right* : l'élément est posé sur le bord droit de son conteneur et le flux courant se déverse sur la gauche du flottant.

- *none* : l'objet ne sera pas flottant.

Soit le flux courant dépasse naturellement la hauteur de l'élément flottant, soit la propriété ***clear*** permet, à un élément, d'anticiper le passage sous l'élément flottant.

G. Formatage de tableaux :

table-layout

Valeur par défaut : `auto` Héritage : `no`

Par défaut, les navigateurs choisissent leur manière d'afficher un tableau : ils décident de l'espace accorder à chaque largeur de colonne, à chaque hauteur de ligne.

Cette propriété permet d'influencer les choix faits par le navigateur. Voici les deux valeurs que peut prendre cette propriété :

- *fixed* : l'espace horizontal laissé pour chaque colonne, ne dépend pas du contenu. Voici la règle suivie :
  - Une définition de colonne fixant la largeur de la colonne est respectée.
  - La propriété ***width*** affectant une cellule de la première ligne est respectée *(quitte à autoriser un dépassement du contenu)*.
  - L'espace restant est équitablement réparti entre les autres colonnes.

  Pour utiliser cette valeur, la largeur ***width*** du tableau doit être explicitement mentionnée même si celle-ci peut être dépassée par le navigateur.
- *auto* : la largeur de chaque colonne est définie par la largeur des cellules qu'elle contient. Le navigateur va ainsi calculer une largeur normale *(sans coupure de ligne)* et une largeur minimale de l'élément pour connaître les largeurs possibles de chaque colonne. Si une cellule possède la propriété ***width***, alors la cellule ne peut avoir une taille inférieure à celle-ci.
  Ensuite lors de la construction du tableau, si celui-ci possède la propriété ***width*** définie par une longueur, alors le navigateur fera de cette longueur la largeur minimale du tableau *(quitte à partager l'espace supplémentaire entre les colonnes ne possédant de propriété **width**)*. Sinon, la largeur du tableau sera calculée au minimum évitant aussi de dépasser la taille de l'élément parent.

Voici un exemple pour illustrer cette propriété :

```
1 <table border=1 style="table-layout:fixed;width:400px"↩
    >
2 <tr><td >1<td>2<td>3<td style="width:25px">4
3 <tr><td style="width:5px">5<td>6<td>7<td>8
4 <tr><td style="width:250px">9<td>10<td>1111<td>12
```

```
5  </table>
6  <p>
7  <table border=1 style="table-layout:auto;width:400px;white-space:nowrap">
8  <tr><td >1<td>2<td>3<td style="width:25px">4
9  <tr><td style="width:5px">5<td>6<td>7<td>8
10 <tr><td style="width:250px">9<td>10<td>1111<td>12
11 </table>
```

Voici le résultat affiché par un navigateur :

1	2	3	4
5	6	7	8
9	10	1111	12

1	2	3	4
5	6	7	8
9	10	1111	12

border-collapse

Valeur par défaut : `separate` Héritage : `oui`

Cette propriété s'applique à l'élément table, elle détermine comment sont dessinées les bordures des cellules. Voici les deux valeurs prises par cette propriété :

- *collapse* : les cellules adjacentes partagent la même bordure.
- *separate* : chaque cellule possède ses propres bordures.

border-spacing

Valeur par défaut : `0` Héritage : `oui`

Dans le cas où la propriété ***border-collapse*** a pour valeur *separate*, cette propriété définit l'espacement entre les bordures de cellules adjacentes. Elle accepte comme valeur :

- une *longueur* : elle définit l'espacement vertical et horizontal entre les bordures de cellules adjacentes.
- deux *longueurs* (voir en appendice) : la première valeur définit l'espacement horizontal et la seconde l'espacement vertical.

empty-cells

Valeur par défaut : `show` Héritage : `oui`

Cette propriété permet d'afficher ou non les bordures des éléments ayant un contenu vide. Il est possible d'afficher ou de cacher la bordure et l'arrière plan d'une cellule

lorsque une cellule est vide. Les deux valeurs possibles de cette propriété sont :
show ; *hide*

Voici quelques remarques supplémentaires :

- La propriété ***display***, disponible pour presque tous les éléments HTML peut prendre les valeurs *table*, *inline-table*, *table-row-group*, *table-column*, *table-column-group*, *table-header-group*, *table-footer-group*, *table-row*, *table-cell* et *table-caption* modifiant ainsi le "*statut*" d'un élément. Mais en pratique peu de navigateur permettent leur utilisation.

  La valeur *inline-table* permet d'afficher un tableau en ligne *(sans retour à la ligne avant et après le tableau)*.

- L'espacement pour les tableaux

  L'élément table peut recevoir les propriétés de style :

  ⇨ ***margin*** qui définit l'espacement entre le tableau et son extérieur.

  ⇨ ***paddin*** définit l'espacement entre la bordure du tableau et l'ensemble des cellules. Mais attention pas entre les cellules.

  L'espacement intérieur d'une cellule peut être modifié avec la propriété ***padding***. Par contre, l'espacement entre les cellules n'est modifiable qu'avec la propriété ***border-spacing*** appliquée à l'élément table.

- L'arrière plan d'un tableau :

  Pour comprendre lorsque plusieurs couleurs d'arrière plan sont appliqués sur un même tableau à différents éléments, il faut imaginer un tableau composé de 6 couches dont celle du dessus est celle des cellules :

Premier plan	• Les cellules • Les lignes • Le groupement de lignes • Les colonnes • Le groupement de colonnes
Arrière plan	• Le tableau

  Ainsi, la couleur de fond d'une cellule sera toujours visible.

H. Autres :

- Le curseur :

cursor Valeur par défaut : `auto` Héritage : `oui`

Cette propriété permet d'affecter une apparence particulière au cuseur lorsque celui-ci

survole l'élément courant.

Voici les valeurs prises par cet élément :

- Les valeurs prédéfinies :

  auto ; *crosshair* ; *default* ; *pointer* ; *move*
  e-resize ; *ne-resize* ; *n-resize* ; *nw-resize* ; *w-resize*
  sw-resize ; *s-resize* ; *se-resize* ; *text* ; *wait*
  help ; *progress*

- Il est également possible de fournir une image en indiquant à cette propriété l'url de celle-ci *(voir en appendice)*.

- Normalement les feuilles de styles apportent un formatage aux éléments HTML, mais n'en modifie pas leurs contenues. La propriété ***content*** permet de rajouter du texte en début ou en fin de contenu d'un élément. Son utilisation est assez complexe et nécessite la lecture de la suite de ce tutorial *(notamment la désignation des éléments par les feuilles de styles - les sélecteurs)*

content

Valeur par défaut : `chaîne vide` Héritage : `non`

Cette propriété n'agit que sur les pseudo-élément *(voir plus bas)* et les éléments munis de la propriété "***display***: *marker*".

Elle ajoute en début ou en fin du contenu de l'élément, le texte passé à la propriété.

- une chaîne vide : cette propriété n'a aucun effet,
- une chaine de caractères non-vide : celle-ci est rajoutée au contenu de l'élément.
- une *URL* : à l'aide de l'écriture fonctionnelle *"url()"*, on ajoute un objet extérieur au contenu de l'élément.
- un compteur pour la valeur *count()*,
- des guillements avec *open-quote*, *close-quote*, *no-open-quote* et *no-close-quote*.

Ainsi, la règle suivante permet d'ajouter à tous les éléments div de la page, le mot "*Monsieur*" en début de contnu.

div:before{***content***:*"Monsieur"*}

- Microsoft propose des filtres modifiant l'apparence des textes, mais ces propriétés ne sont valables que pour Internet Explorer : ce ne sont pas des standarts *(cherchez sur internet pour davantage de renseignement)*.

I. Les listes :

- Les propriétés suivantes s'utilisent pour modifier l'apparence des listes, elles ne peuvent s'utiliser que sur des éléments ayant la définition de style :

  "***display***: *list-item*"

list-style-type

Valeur par défaut : `disc` Héritage : `oui`

Cette propriété détermine le type de puce placé avant chaque élément de la liste. Les valeurs prises se passent de commentaires :

disc ; *circle* ; *square* ; *decimal* ; *decimal-leading-zero* ; *lower-roman* ; *upper-roman* ; *lower-greek* ; *lower-latin* ; *upper-latin* ; *armenian* ; *georgian* ; *lower-alpha* *upper-alpha* ; *none*

list-style-image

Valeur par défaut : `none` Héritage : `oui`

Cette propriété permet d'utiliser une image à la place des puces proposées par défaut. Elle s'utilise avec la notation fonctionnelle *url() (voir en appendice)*

list-style-position

Valeur par défaut : `outside` Héritage : `oui`

Cette propriété spécifie l'emplacement des puces relativement au contenu de la liste :

- *outside* : la puce est placée à l'extérieur de l'élément de la liste.
- *inside* : la puce est placée à l'intérieur de l'élément de la liste, ça en sera son premier élément.

liste-style

Valeur par défaut : `voir chaque propriété` Héritage : `oui`

Cette commande rassemble les trois propriétés ***list-style-type***, ***list-style-position***, ***liststyle-image*** en une seule propriété. La valeur fournit à cette propriété est une ou plusieurs des valeurs des sous-propriétés séparées par des espaces.

- Sans rentrer dans les détails car ce passage deviendrait trop technique, il est possible de créer des compteurs directements à l'aide des styles CSS et de les contrôler.

  ⇨ On crée les compteurs à l'aide de la propriété ***content*** utilisée sur des pseudo-éléments muni de `:before` à l'aide de la fonction "*counter()*"

⇨ Voici les propriétés de styles alors accessibles :

counter-reset

Valeur par défaut : `none` Héritage : `non`

Cette propriété réinitialise un compteur. Voici les deux formes d'utilisation de cette propriété :

- "***counter-reset***: *chaine*"
  Le compteur identifié par *chaine* sera réinitialisé à 0.
- "***counter-reset***: *chaine nbr*"
  Le compteur identifié par *chaine* sera réinitialisé à partir de *nbr*.

counter-increment

Valeur par défaut : `none` Héritage : `non`

Cette propriété permet de modifier l'incrémentation d'un compteur : elle définit le nombre qui sera rajouté au compteur chaque fois qu'un nouveau élément de la liste est rencontré. Voici les deux manières d'utiliser cette propriété :

- "***counter-increment***: *chaine*"
  Le compteur *chaine* aura une incrémentation sera d'une unité.
- "***counter-increment***: *chaine nbr*"
  Le compteur identifié par *chaine* aura une incrémentation de *nbr*.

III. Affectations des styles aux éléments HTML :

A. Généralités et principes de bases :

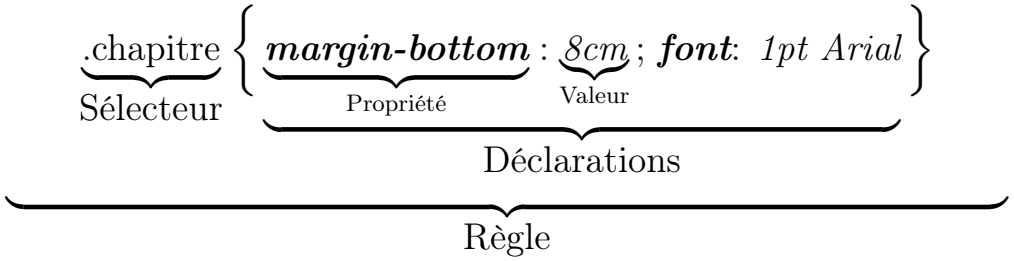
Il existe deux manières d'affecter des propriétés de styles à un élément :

- Les déclarations de style sont directement écrites dans la balise ouvrante de l'élément à l'aide de l'attribut `style`.
- Les déclarations de style peuvent être écrites extérieurement à l'élément ; on définie alors une *règle de style* contenant ces déclarations. Voici deux manières de créer des règles de styles :
  - dans une feuille de style interne : les règles de styles sont placées dans un élément style qui lui-même doit se trouver dans l'entête du document head.
  - ou dans une feuille externe : les règles de styles sont placés dans un fichier externe à la page ; ce dernier est lié à la page Web à l'aide de l'élément link.

Dans le prochain chapitre, nous étudierons plus précisemment comment mettre en place chacune de ces trois façons d'utiliser les styles CSS. Intéressons nous comment une règle de style

va être rattachée à un élément de la page ; cet élément récupére alors les déclaration de styles de la règle.

Voici la structure d'une règle de style définie extérieurement à l'élément :



Une règle se décompose en deux parties principales :

- Le sélecteur : il permet de déterminer le ou les éléments de la page qui recevront les déclaration de cette règle.

- La déclaration : elle contient les différentes déclarations de styles ; chacune de ces déclarations sont séparées par un point virgule.
  Une déclaration de style est la donnée d'un couple $^{\textbf{\textit{propriété}}}/_{valeur}$, elle s'écrira sous la forme :
  propriété: *valeur*

B. Première étude du sélecteur :

Voici l'extrait d'un fichier définissant trois règles de styles :

```
1  Affecte tous les éléments "div"
2    div{font: 10px; margin: 10px 5px; padding:5px;}
3
4  Affecte les éléments de classe "chapitre"
5    .chapitre{font:18pt Arial; margin:8px 0px}
6
7  Affecte les éléments d'identifiant "titre"
8    #titre{font:24pt "Comic Sans MS"}
```

Etudions le sélecteur :

- Le nom du sélecteur n'est précédé d'aucun caractère :
  Cette règle va s'appliquer à tous les éléments HTML de la page ayant le même nom que le sélecteur. Dans l'exemple, tous les éléments div auront leur contenu avec une fonte de taille 10 points.

- Le nom du sélecteur est précédé d'un point :
  Cette règle s'applique à tous les éléments HTML dont l'attribut `class` a pour valeur le nom du sélecteur. Ainsi, dans l'exemple, tous les éléments de la page ayant la balise ouvrante de la forme :
  `<bal class="chapitre">`
  auront leur contenu affiché avec une fonte de 18 points.

- Le nom du sélecteur est précédé d'un dièse :

  La règle s'applique au seul élément de la page ayant pour valeur de l'attribut `id` le nom du sélecteur. Dans l'exemple, l'élément de la page courante, s'il existe, ayant la balise ouvrante de la forme :

  `<bal id="titre">`

  recevra la déclaration de style de cette règle ; son contenu est affiché avec une fonte de 24 points.

Au vu de cet exemple, une question naturelle peut se poser :

`Que se passe-t-il si un élément est de nature div et affecté de l'attribut class avec la valeur chapitre?`

Dans ce cas, l'élément reçoit les deux déclarations : voici pourquoi les styles CSS s'appelle les "*feuilles de styles en cascades*".

Mais une question se pose, dans le cas de l'exemple, cet élément recevra une fonte de taille 10 ou 18 points? Nous verrons plus loin, que dans le cas de conflit, comment le navigateur choisit la déclaration à utiliser.

C. La désignation des éléments :

La manière d'écrire un sélecteur peut être nettement plus compliqué que précédemment ; on peut alors sélectionner plus "*finement*" les éléments d'une page qui seront liés à une règle.

Actuellement, FireFox prend en compte l'ensemble de ces sélecteurs ; Internet Explorer n'en prend en compte que trop peu.

1. *Principes de bases :*

*	Désigne tous les éléments HTML. Cet élément *(appelé le sélecteur universel)* est implicite, il peut être ommis.
E	Désigne tous les éléments HTML de type *E (par exemple body, br, h1 . . .)* de la page
* *.maClasse* ou *.maClasse*	Désigne tous les éléments HTML dont l'attribut `class` contient la chaîne *maClasse*.
* *#unId* ou *#unId*	Désigne tous les éléments HTML qui possède l'attribut HTML `id` avec la valeur *unId*.

2. *Désignation par filiation :*

E et *F* désigne des sélecteurs quelconque : ils désignent donc des éléments de la page. Voici les différentes compositions possibles :

E F	Désigne les éléments sélectionnés par *F* qui appartiennent à un élément sélectionné par *E*. C'est à dire les éléments *F* qui soient contenus dans les éléments *E*.
E > *F*	Désigne les éléments sélectionnés par *F* *directement* contenus dans un élément sélectionné par *E* : exemple les fils mais pas les petits fils
E + *F*	Désigne tous les éléments désignés par *F* étant directement précédé d'un élément désigné par le sélecteur *E*

3. *Désignation à l'aide des attributs de l'élément :*

E[*foo*]	Désigne tous les éléments désigés par le sélecteur *E* possédant l'attribut *foo*.
E[*border*="*thin*"]	Désigne tous les éléments désignés par *E* dont la propriété ***border*** est définie avec la valeur *thin*. *E*[*id*=*monId*] est équivalent à *E*#*monId*
E[*border*~="*thin*"]	Désigne tous les éléments sélectionnés par *E* dont la valeur de la propriété ***border*** contient la valeur *thin* séparée par des espaces. div[*class*~="couleur"] sélectionne l'élément dont la balise ouvrante est : `<div class="couleur fond">`
E[*lang*|="*en*"]	Va sélectionner par exemples tous les éléments dont la valeur de l'atribut *lang* commence par les deux lettres *en*.

D. Les pseudo-classes et les pseudo-éléments :

L'essentiel du langage CSS est de désigner des éléments et d'apporter un formatage à l'ensemble de leurs contenus. Nous allons voir certaines propriétés de styles qui ont un comportement particulier.

E désigne un sélecteur et par là, on représente également par *E* un élément sélectionné par ce selecteur.

1. *Les pseudo-éléments :*

- On peut rajouter du contenu à un élément à l'aide d'une du sélecteur :

  E:after

  La propriété ***content*** permet de rajouter du contenu en fin d'élément. Le reste de la déclaration gèrera le style du contenu ajouté.

- La première lettre d'un élément est désigné par le sélecteur :
  E:first-letter
  La déclaration de style d'une tel règle appartera un formatage particulié à la première lettre de *E*.

- On peut apporter également un formatage particulier au premier élément HTML contenu dans l'élément *E* à l'aide du sélecteur :
  E:first-child

2. *Les pseudo-classes :*

Les pseudo-classes permettent de sélectionner les éléments lorsque ceux-ci reçoive une attention particulière du navigateur :

E:active	Ce sélecteur appliquera la déclaration de style de la règle lorsque le client enfoncera le bouton de la souris sur l'élément *E*.
E:hover	La déclaration de style de cette règle s'appliquera à l'élément *E* lorsque le curseur de la souris survolera l'élément.
E:focus	Ce sélecteur agira lorsque l'élément *E* reçoit le focus.

Plus particulièrement, voici deux pseudo-classes liées aux liens :

a:link	Ce sélecteur désigne un lien qui n'a pas été visité.
a:visited	Ce sélecteur désigne un lien déjà visité.

IV. L'inclusion des définitions de styles :

A. Présentation :

Il existe trois manières d'intégrer les définitions de styles CSS :

1. Les ***styles internes*** :
   On place directement les déclarations de styles dans la balise d'ouverture d'un élément en utilisant l'attribut HTML *style* ; la valeur de cet attribut sera la déclaration de style appliquée à l'élément.

```
1 <body style="background-color:#6666FF">
2  Premier exemple :
3  <div style="border: 2pt dotted green;font-size:large↩
       ">
4    de définitions de style.
5  </div>
```

```
</body>
```

2. Les ***feuilles de styles internes*** :

Les règles de style sont placés dans l'entête head de la page à l'intérieur de l'élément style :

```
<head>
 <style  type="text/css">
   .chapitre{font-size:10pt;color:red}
   #titre{font-size:x-large;font-style:italic}
 </style>
</head>
```

Identifiez le contenu de la balise :

`<style type="text/css">...</style>`

3. Les ***feuilles de styles externes*** :

Les règles de styles sont directement écrites dans un fichier externe *(généralement d'extension CSS)*. On relit une feuille de style externe à une page Web, en insérant l'élément link dans l'entête de la page.

link

```
<head>
 ...
 <link  href="urlFeuille.css"  rel="stylesheet"  type="text/css">
 ...
 </head>
```

La feuille de style externe `urlFeuille.css` est un simple fichier texte contenant des règles de styles *(une règle par ligne)*. Voici un exemple d'un tel fichier :

```
.chapitre{font-size:10pt;color:red}
#titre{font-size:x-large;font-style:italic}
```

B. Comparaisons des différents modes d'insertion :

Voici une comparaison de ces méthodes :

- La méthode 1. est utilisée pour adjoindre, occasionnellement, un style à un élément.
  Il présente l'inconvénient d'alourdir le code HTML : pour ajouter les mêmes déclarations de style à un même élément, il est préférable d'utiliser une des deux autres méthodes.

- Les méthodes 2. et 3. permettent de définir les règles de styles extérieurement aux éléments, ainsi une même règle peut être utilisée simultanemment à plusieurs éléments.

La méthode 3. est généralement utilisée lorsqu'une règle de style est utilisée par des éléments de différentes pages.
La méthode 2. est utilisée pour adjoindre des règles supplémentaires à une page sans avoir besoin d'en faire profiter d'autres pages du site.

Il existe plusieurs avantages à utiliser une feuille externe de styles :

- une même feuille de styles peut être utilisée par plusieurs pages : ainsi, on peut aider à l'armonisation du formatage de l'ensemble d'un site. Ceci facilitant également la maintenance des styles.
- si les navigateurs mettent en cache les feuilles de style, le client retournant sur un site n'aura pas besoin de la télécharger de nouveau.

C. L'élément link :

Voici les différentes caractéristiques de l'élément link :

*<**link** rel='stylesheet' type='text/css' href='m' media='n' title='p'>*

Voici les différentes attributs de cette balise utile pour l'insertion d'une feuille de styles :

- `type` *doit avoir la valeur text/css.*
- `href` *doit contenir l'url, relative ou absolue, de la feuille externe de style.*
- `media` *permettra d'utiliser des définitions de styles par exemple pour l'affichage à l'écran et d'autres pour l'impression (voir en appendice).*
- *Certains navigateurs (tel qu'Opera) permettent l'utilisation de plusieurs feuilles de styles pour une même page, le client choisira une feuille de style à l'aide de son intitulé défini par* `title`*.*

V. Priorités en cascade sur les feuilles de styles :

A. Le conflit des règles :

Un élément peut recevoir de plusieurs manières des déclarations de styles :

- par des styles internes ;
- par des feuilles de styles, internes ou externes ; dans ce cas, il recevra des règles de styles en fonction de son sélecteur : nature de la balise, attribut `class`, attribut `id`...

On peut facilement concevoir l'exemple d'un élément recevant des règles par différents biais et par là obtenir différentes valeurs d'une même déclaration de styles :

```
<head>
 <style>
   div{font-size:8pt}
   .chapitre{font-size:10pt}
   #important{color:red}
 </style>
<head>

<body>
 <div  id="important"  class="chapitre"
       style="font-style:12pt">
   ....
 </div>
</body>
```

Ainsi, l'élément div reçoit alors trois déclarations du style ***font-size***. Nous allons voir, dans ce paragraphe, comment les navigateurs prennent la décision d'utiliser une déclaration plutôt qu'une autre.

B. Les différentes feuilles de styles externes :

Nous avons vu comment liée une feuille de style externe à une page à l'aide de l'élément link.

Mais les navigateurs peuvent inclure également d'autres feuilles de styles :

- Certains navigateurs *(Actuellement Opéra et FireFox)* permettent au client de rajouter lui-même des feuilles de styles.
  Ceci peut permettre, par exemple, au mal voyant d'augmenter la taille des caractères.

- Le navigateur possède lui-même des feuilles de styles par défaut définissant la valeur par défaut des propriétés de styles : taille des caractères, polices utilisés...

Ainsi, un élément HTML peut recevoir pour une propriété de style plusieurs valeurs venant de sources variées.

C. Le média de sortie :

Il est possible d'afficher une page avec le fond noir et le texte en blanc mais lorsqu'il s'imprimera le texte sera en noir sur fond blanc.

Il est possible de définir des règles qui ne seront utilisés que pour un certain type de média :

Voici un exemple du contenu d'une feuille de style externe régionant son contenu suivant le média utilisé :

```
Ici, règles de styles communes à tous les médias
```

```
2
3  @media print {
4  Ici, règles pour les imprimantes
5  }
6
7  @media screen {
8  Ici, règles pour l'affichage à l'écran
9  }
10
11 @media screen, print {
12 Ici, règles communes pour l'écran et l'impression
13   }
```

Voici la liste des différentes sorties prises en charge par CSS :

- **all** : représente tous les types de média de sortie.
- **braille** : destine les propriétés de styles pour les machines de braille
- **embossed** : désigne les imprimantes en brailles.
- **handheld** : désigne les appareils possédant un petit écran.
- **print** : désigne les imprimantes
- **projection** : désigne l'affichage de la page en plein écran.
- **screen** : désigne les écrans couleurs.
- **speech** : désigne les synthétisuers de voix.
- **tty** : désigne les médias limités quand à leurs modes d'affichage.
- **tv** : représente les média de type télévision.

D. Les définitions de style ¡importantes! :

Lorsqu'un navigateur reçoit plusieurs valeurs pour un même style, il est possible d'infléchir la position du navigateur à l'aide du mot-clef *important*. Voici un exemple :

```
1 <head>
2  <style>
3    .ii{color:yellow ! important}
4    #ee{color:red}
5  </style>
6 </head>
7 <body>
8  <div id=ee class=ii>ffsdfsdf</div>
```

```
9 </body>
```

L'élément div reçoit deux valeurs pour la propriété ***color***, la valeur *yellow* sera ici utilisée grâce au mot-clef *! important*.

E. Ordre de sélection :

Voici les mécanismes qui permettent au navigateur de sélectionner une déclaration plutôt qu'une autre en cas de conflit.

Gardez bien en mémoire qu'en cas de conflit, le navigateur va comparer ses déclarations au crible des quatres règles ci-dessous, dès qu'il en restera qu'une, c'est celle qui sera appliquée à l'élément :

1. **Récupération des valeurs en fonction du média :**

   Le navigateur sélectionne seulement les règles adaptées au média utilisé.

2. **Tri par importance et provenance :**

   Puis, voici dans l'ordre décroissant d'importance, les critères de sélection d'une déclaration en fonction de sa provenance et de la présence ou non du mot-clef *! important*.

   a. Style important de l'auteur *(du créateur de la page)*,

   b. Style important de l'utilisateur *(du client)*,

   c. Style normal de l'auteur,

   d. Style normal de l'utilisateur,

   e. Style par défaut du navigateur *(ie navigateur)*.

3. **Tri par spécificité du sélecteur:**

   La nature du sélecteur essayant d'attribuer une propriété est importante. Plus le sélecteur est précis, plus sa déclaration sera retenue au dépend des sélecteurs moins précis :

   En simplifiant, voici par ordre décroissant l'ordre du choix de la propriété en fonction de la nature du sélecteur :

   a. L'attribut *style* des éléments HTML,

   b. Le sélecteur est l'attribut *id*,

   c. Le sélecteur est l'attribut *class*,

   d. Le sélecteur est le type de l'élément.

   L'exemple suivant illuste ce point :

```
1 <head>
2  <style>
3   div {margin:0pt; color:red; font-size:10pt; border:↩
       solid red;padding:0px:margin:0px}
```

```
4     .maClasse {color:blue ! important; font-size:20pt;↩
          padding:0px;margin:10px}
5     #monId {color:green;border: solid green;padding:0px}
6   </style>
7 </head>
8 <body>
9   <div id="monId" class="maClasse" style="padding:10↩
        px">
10    Mon conteneur div
11  </div>
12 </body>
```

La définition de styles pour l'élément div sera :

{***margin***:*10px* ; ***padding***:*10px* ; ***color***:*blue* ; ***border***:*solid green* ;}

4. **Tri par ordre de définition :**

Si les critères précédents n'ont pas suffit à déterminer la valeur à appliquer à la propriété d'un élément alors le navigateur utilisera la dernière déclaration rencontrée.

VI. Annexe :

A. Les URL dans les feuilles de styles :

Certaines propriétés de styles prennent comme argument un fichier externe *(par exemple une image)*. Pour indiquer l'URL de ce fichier, on utilise la fonction "*url(...)*".

Voici un exemple de la propriété ***background*** affichant une image dans l'arrière-plan d'un élément :

background: *url("http://www.exemple.fr/image.png")*

Dans le cas d'utilisation d'URL relatives dans des feuilles externes de styles, l'adresse de départ est celle de la feuille de style et non pas celle de la page courante.

à

B. La valeur inherit :

Toutes les propriétés de styles *(ou presque)* acceptent la valeur *inherit* : cette valeur permet à la propriété de récupérer la valeur de l'élément parent.

La valeur par défaut de la propriété ***border*** est "*none*". Voilà un exemple d'utilisation de la valeur *inherit* :

```
1 <div style='border: red solid;padding:0.2cm'>
2  <span>Première boîte</span>
3  <span style='border:inherit '>Seconde boîte</span>
4 </div>
```

Le second span possède la même bordure que son élément parent div ; par contre, le premier div a la propriété ***border-style*** avec la valeur *none*.

C. Les couleurs dans les feuilles de styles :

Pour une meilleure compréhension des couleurs utilisées dans le Web, reportez-vous à l'annexe de la formation sur le langage HTML :

- Les couleurs *(au format sRGB* correspond à un nombre héxadécimal de 6 chiffres *(3 octets correspondant à chaque composante additive Red, Green, Blue)* sous la forme :

  #RRGGBB

- Le format CSS comprend 17 couleurs prédéfinies :

  aqua, *black*, *blue*, *fuchsia*, *gray*, *green*, *lime*, *maroon*, *navy*, *olive*, *orange*, *purple*, *red*, *silver*, *teal*, *white* et *yellow*

  et il utilise également les couleurs prédéfinies dans le format HTML

- La notation fonctionnelle "*rgb(a,b,c)*", utilisable dans les déclarations, permet d'obtenir

une couleur où chacune des trois valeurs correspondants respectivement à la quantité de rouge, de vert, de bleu composant la couleur.

Voici la nature des nombres a, b et c :

- des entiers en base décimale compris entre 0 et 255 ;
- des pourcentages de 0% à 100%.

D. Les longueurs :

Voici les unités de longueur utilisées dans les pages Web classés en deux types de longueurs

- Les "*unités absolues*" sont des unités indépendantes :
  inches *(in)* ; centimètre *(cm)* ; millimètres *(mm)* ; point *(pt)* ; pica *(pc)*.
- Les "*unités relatives*" définissent une unité relativement à la fonte courante ou de la résolution du média de sortie :
  - L'unité **em** représente la largeur de la lettre minuscule "m" dans la fonte courante.
  - L'unité **ex** représente la hauteur de la lettre minuscule "x".
  - L'unité **px** dépend du média de sortie et de la résolution de rendu de celui-ci *(par exemple résolution d'impression de l'imprimante)*.

Si une longueur est attendue et que seul un nombre est présent, alors l'unité pixel sera utilisée.

E. La règle @import :

La règle *@import* permet d'importer les déclarations de styles contenues dans un fichier externe. Voici différentes utilisations de cette règle :

- *@import "mesStyles.css";* ou *@import url("mesStyles.css");*
  Ces déclarations permettent d'importer les définitions de styles contenues dans le fichier "mesStyles.css"

Cela peut permettre d'inclure des règles de styles directement dans un média :

- *@import url("mesStyles.css") print;*
  Cette déclaration permet d'importer les définitions de styles contenues dans le fichier "mesStyles.css" mais elles ne seront importées que pour l'imprimante comme média.
- *@import url("mesStyles.css") projection, tv;*
  Ici l'importation des définitions de styles seront utilisés pour les médias *projection* et *tv*.


Cette règle doit précéder toutes autres règles dans une feuille de style.

Index