



[Date]

Support de cours

CSS

Enseignant
OUSMANE A. MATINE

CHAP4 : LES FEUILLES DE STYLES CSS

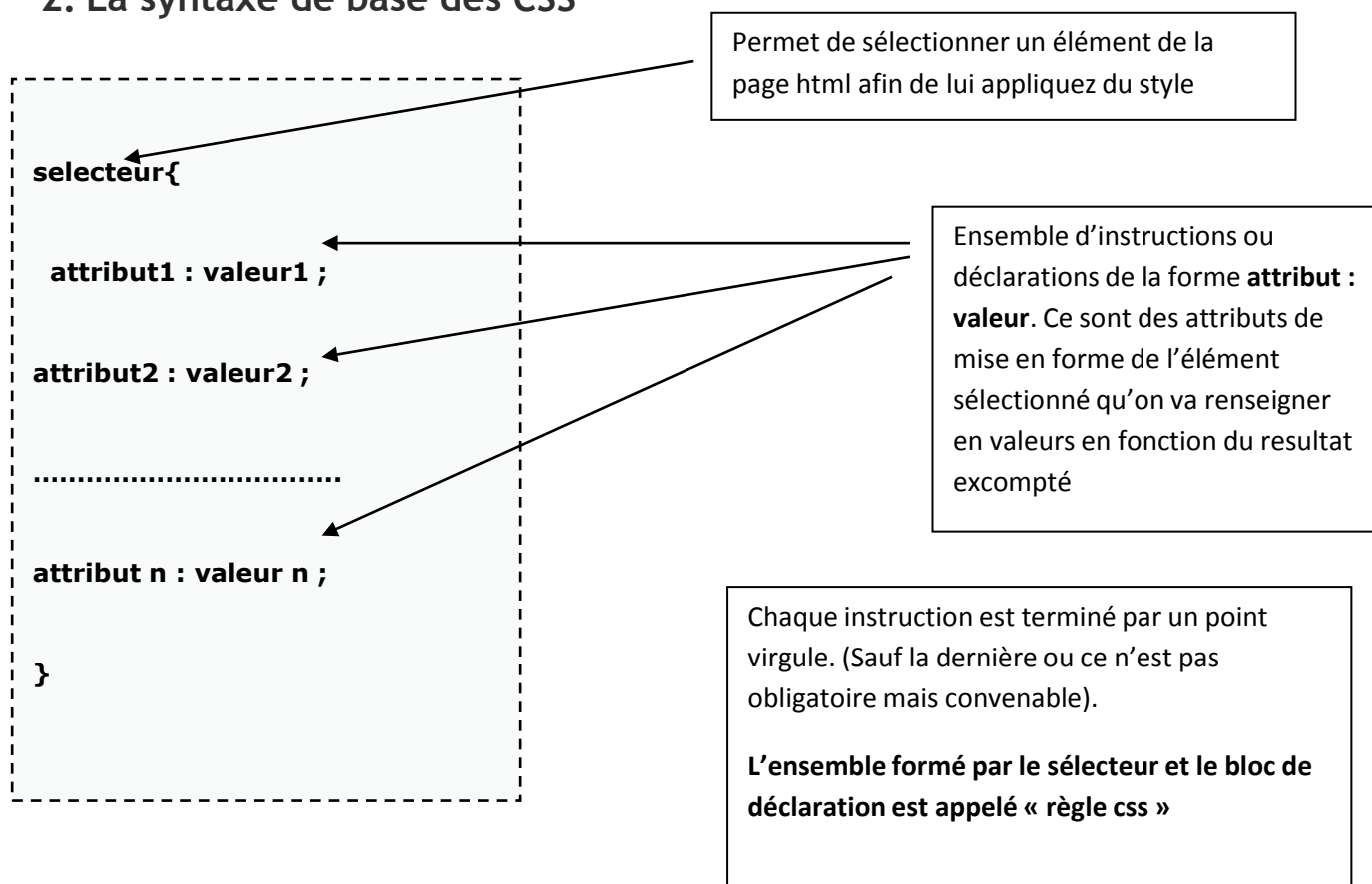
Créées pour prendre en charge tous les aspects graphiques et les rendus sur les différents médias (écran, mais aussi imprimante, synthèse vocale, assistant personnel, etc.), les feuilles de styles ajoutent la couche graphique au document web et à sa structure. Cette partie aborde leur syntaxe et utilisation pratique allant de la typographie aux différentes méthodes de positionnement.

1. Intérêts du ccs

Rappelons que les normes du Web incitent désormais webmasters et concepteurs de sites web à séparer clairement leurs contenus (HTML) de la mise en forme (CSS). Cette distinction un intérêt principal, la conception des pages web s'en trouve vraiment facilitée :

- Cette dissociation des fonctions évite encore de pénaliser les anciens navigateurs (les documents y demeurant lisibles), les navigateurs en mode texte (ciblant les aveugles ou les utilisateurs en mode texte), et les programmes utilisant d'autres médias (assistants personnels, WAP, ordinateurs avec synthétiseur vocal, navigateurs braille, etc.).
- Quel webmaster n'a jamais dû modifier tout son site, page après page, pour répéter fastidieusement la même manipulation élémentaire (par exemple, changer la couleur des titres) ? CSS simplifie cette opération : avec cette nouvelle technique, une seule feuille de styles, stockée dans un fichier, assure la gestion graphique de l'intégralité d'un site, qu'il compte trois pages ou plusieurs centaines. Toute intervention dans ce fichier sera immédiatement répercutée partout.
- En outre, cette feuille CSS étant conservée dans la mémoire cache de l'ordinateur du navigateur après la première connexion, toutes les pages du site s'afficheront plus rapidement que si les indications de présentation étaient répétées sur chacune d'entre elles.

2. La syntaxe de base des CSS



Exemple :

```
h1
{
    color :blue ;
    tex-align : center;
}
```

Cette règle colorera en bleu le contenu texte (ici titre de niveau 1) des balises <h1> contenues dans les pages auxquelles ce fichier css sera lié.

Elle centrera aussi les titres en par rapport à l'espace alloué aux éléments <h1>

3. Appliquer les styles CSS

Comment appliquer ce style CSS au document HTML qui en dépend ? Il existe pour cela trois méthodes, mais nous utiliserons de préférence celle de la feuille de styles, qui présente de nombreux avantages.

➤ Insérer des styles dans les balises

Ici on insère directement(ou localement) dans le code html, le style sur une balise à l'aide de l'attribut **style**. Voici un exemple :

```
<h1 style="color :blue ; text-align :center ;" >Mon Titre</h1>
```

Cette technique ne respectant pas la philosophie des normes xhtml/css(séparer le contenu et le design), nous garderons de l'utiliser.

➤ Insérer des styles dans l'en-tête du document

On peut d'abord placer des styles CSS dans l'en-tête HTML (contenu de l'élément<head></head>). Placé entre les balises **<style>** **</style>**, ce code s'appliquera à tout le document (doc a toutes les balises <h1> du document):

```
<style type="text/css">
h1 {color: blue; tex-align : center;}

</style>
```

Dans cette technique, qui sépare correctement contenu et mise en forme, malheureusement, la portée de ce style se limite au document HTML du fichier. Dans l'idéal, le design général du site s'appliquera automatiquement, sans devoir être explicité dans chaque document HTML. Pour aboutir à cet effet, **nous placerons les règles CSS dans un fichier distinct.**

➤ Lier les styles à partir d'une feuille séparée

Il s'agit de stocker les ressources dans des fichiers distincts : documents HTML et feuilles de styles CSS. Ces dernières renfermeront toutes les règles nécessaires à la mise en page et au design des fichiers HTML. Il suffira alors de modifier le fichier CSS séparé pour changer l'allure de toutes les pages HTML du site. Ce fichier CSS, appelé feuille de styles, portera l'extension .css et ne contiendra que des règles CSS (aucun code HTML n'y sera autorisé, pas même la balise <style> vue dans la méthode précédente). Le langage CSS, fait pour écrire des règles, ne peut contenir d'autres langages. Il convient ensuite de relier ce fichier aux contenus sur lesquels il est censé porter.

✓ Liaison par la balise <link>

Pour lier cette feuille de styles à toutes les pages HTML du site, il est d'usage de placer une balise <link> dans l'en-tête (<head>) de ces dernières.

Fichier html à auquel est lié la feuille de style de nom : styles.css

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<title>Titre évocateur pour le document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-15" />
<link rel="stylesheet" type="text/css" href="styles.css" media="all" />
</head>
<body>
<h1>Titre principal de la page</h1>
</body>
</html>
```

La feuille de styles CSS styles.css aura alors pour contenu :

```
h1 {
  color: blue;
  text-align : center;
}
```

✓ Utiliser la règle @import

Cette règle permet elle aussi de lier une feuille de styles externe à son document HTML. Ce n'est pas une balise HTML, mais une règle CSS 2. Il est donc nécessaire de la déclarer dans l'élément <style> de l'en-tête du document :

```
<style type="text/css">
@import url(styles.css);
</style>
```

✓ MÉTHODE link ou @import ?

Quelle méthode est préférable pour relier la feuille de styles au document ? Cette question fréquente mérite un développement.

<link> est une balise HTML qui n'est pas uniquement dévolue aux feuilles de styles. Quand elle désigne une feuille de styles CSS, elle s'accompagne des propriétés et valeurs suivantes : rel="stylesheet", type="text/css" et media=[type demédia souhaité], voire title dans le cas de feuilles de styles persistantes et alternatives.

Par exemple :

```
<link rel="stylesheet" href="/styles/habillage.css" type="text/css" media="screen" />
```

La règle @import, propriété CSS 2, sera suivie de l'URL d'un fichier contenant les styles à appliquer en plus de la feuille de styles en cours. On pourra préciser une liste de médias. Par exemple :

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

```
@import url(/styles/habillage.css)
```

```
;</style>
```

ou :

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

```
@import url(impression.css) print;
```

```
</style>
```

Cette propriété permet en outre d'inclure des feuilles de styles dans d'autres, ce qui permet de créer des feuilles de styles dynamiques sans devoir recopier plusieurs fois le même code. Le résultat est en pratique identique (ce qu'on vérifie par des tests sur plusieurs sites), à une petite subtilité près. En effet, cette règle CSS 2 n'est pas reconnue par les très anciens navigateurs, pas aux normes CSS – c'est par exemple le cas de Netscape 4. Appliquer une feuille de styles par appel à @import la fera donc appliquer partout sauf sur ces anciens navigateurs, ce qui permettra à leurs utilisateurs de consulter le site web concerné sans trop de problèmes. Un site dépourvu de feuille de styles est toujours plus lisible qu'un site dont les styles sont mal interprétés.

En résumé : avec @import un Netscape 4 recevra une page brute, sans style, plutôt qu'une horreur probablement illisible. C'est donc une technique recommandée pour l'interopérabilité et la compatibilité avec les anciens navigateurs.

4. Les sélecteurs de styles

Dans les exemples ci-dessus on utilise comme sélecteur les balise <h1>. Cette règle affecteront donc toute balise <h1> dans les pages HTML concernées. Cet effet n'est pas toujours désirable : on souhaite parfois n'intervenir que sur certaines balises d'un type donné (par exemple, en colorant en rouge certains liens hypertextes mais pas tous). Pour cela, le langage CSS accepte **différentes formes de sélecteurs** :

- les **sélecteurs de balises** (utilisés jusqu'ici) ;
- les **sélecteurs de classes** (une classe est un nom donné à un ensemble d'éléments HTML à distinguer) ;
- les **sélecteurs d'identifiants** (un identifiant ou id est le nom attribué à un élément unique dans le document HTML) ;
- les **pseudo-classes** et les **pseudo-éléments** (variantes pour certaines fonctionnalités, par exemple les liens).

➤ Les balises

Toute balise HTML peut intervenir dans un sélecteur. Ainsi, on pourra supprimer tous les interlignes entre paragraphes en attribuant à la balise <p> des marges haute et basse nulles :

```
p {  
margin-top: 0;  
margin-bottom: 0;  
}
```

➤ Les classes

Une classe est **un nom** que l'on choisit librement (en se limitant aux caractères alphanumériques classiques) et dont **on baptise les éléments concernés. Un sélecteur de classe reprend son nom en le préfixant d'un point** (par exemple : **.ma_classe**, **.toto**, etc.). Pour attribuer un comportement différent à certains éléments, il suffit de leur appliquer une classe. On affichera en vert les liens hypertextes du document à l'exception de certains liens particuliers que l'on souhaite voir apparaître en rouge en reprenant la technique suivante. Une première règle précise le comportement par défaut à adopter pour toutes les balises <a> :

```
a {  
color: green;  
}  
  
.sommaire {  
color: red;  
}
```

```
<a href=".">lien normal</a>  
  
<a href="." class="sommaire">lien  
rouge</a>
```

➤ Les identificateurs

Un identificateur (identifiant, ou id) est lui aussi un nom choisi librement (en se limitant aux caractères alphanumériques classiques). Il se distingue de la classe en ce qu'il ne peut porter qu'au plus sur un objet du document. Les sélecteurs CSS s'y réfèrent par l'emploi d'un caractère dièse (#) suivi de son nom (exemples : #exemple, #toto, #banniere2, etc.)

```
#special {  
background-color : navy ;  
color :white ;  
}
```

```
<p id= "special">  
  
Ce paragraphe est spécial. Il a une couleur  
de fond bleu marine et une couleur de  
police blanche .  
  
< /p>
```

➤ Les pseudo-classes et les pseudo-éléments

Les pseudo-classes et les pseudo-éléments créent des mécanismes ou des relations qui ne sont pas réalisables en HTML. CSS crée en effet des éléments spécifiques à certaines actions (comme le survol d'un lien) ou à certaines arborescences (comme le premier paragraphe d'un bloc). Ces techniques permettent de styler un contenu n'apparaissant même pas dans le code du document. Un exemple courant est l'utilisation de la pseudo-classe **:hover**, qui prend effet lorsque le pointeur de la souris survole l'élément concerné. Ainsi, la règle suivante

```
a:hover {  
  text-decoration: none;  
}
```

Cette règle affectera tous les liens de la page lors de leur survol par le pointeur de la souris. :

texte du lien

Avec cette instruction, les liens hypertextes – soulignés par défaut – apparaîtront sans ornement lorsque le pointeur de la souris les survolera.

NB :

COMPATIBILITÉ Internet Explorer et la pseudo-classe :hover

La pseudo-classe :hover permet d'affecter un style particulier lorsque l'élément désigné est survolé. A priori, cette pseudo-classe n'est pas réservée à l'élément de lien <a> mais peut être appliquée à tous les éléments ; ainsi est-il possible de modifier la couleur d'arrière-plan d'un bloc entier lors du survol à l'aide de la souris.

Cependant, jusqu'à sa version 6, Internet Explorer ne prend en charge :hover que lorsqu'elle est appliquée à l'élément <a>. IE7 corrige cette lacune et étend son application à tous les éléments.

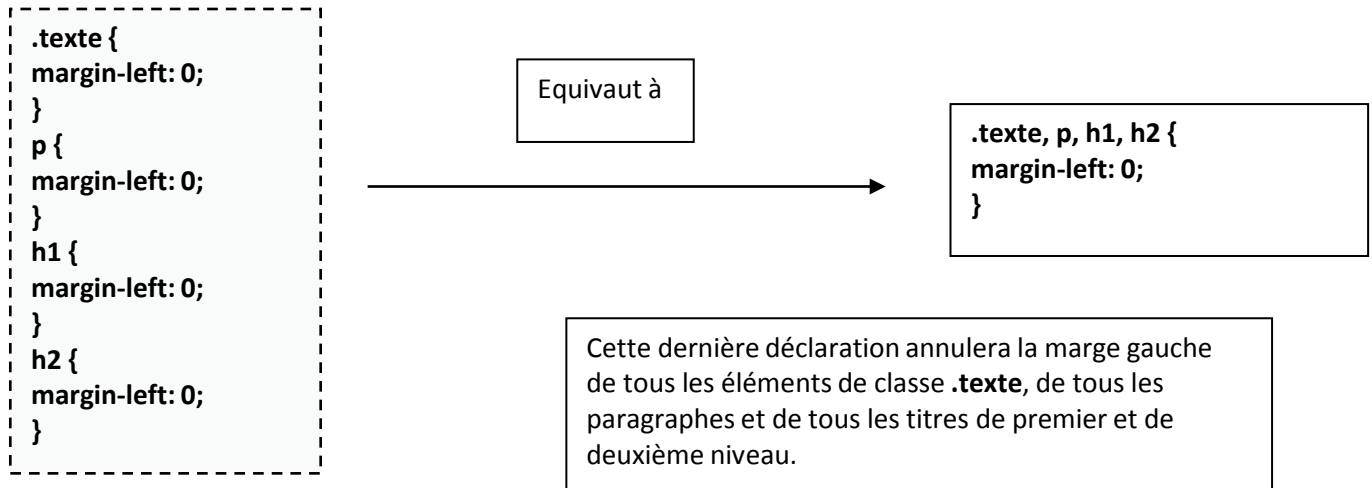
Les pseudo-éléments **:first-letter** et **:first-line** (qu'Internet Explorer ne reconnaît qu'à partir de sa version 6) agissent sur la première lettre ou la première ligne d'un paragraphe, indépendamment de la balise qui structure sinon ce contenu.

5. Syntaxes de regroupements

Certaines règles permettent de faciliter et d'alléger considérablement la production de code CSS. Observer ces principes simples vous fournira des feuilles de styles aérées, bien plus compréhensibles.

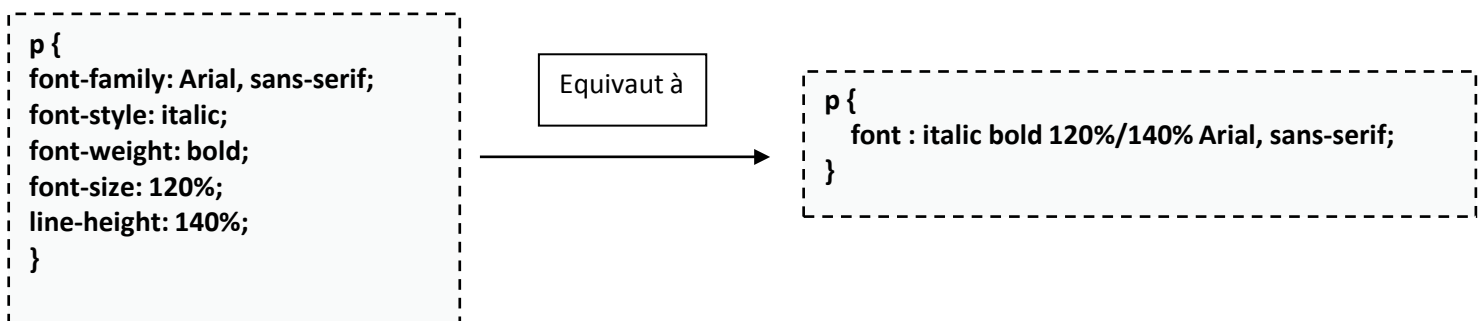
➤ Regroupement des sélecteurs

La première construction syntaxique utile permet de regrouper des sélecteurs. Au lieu de répéter la même règle pour plusieurs éléments, on pourra factoriser leurs sélecteurs en les séparant par des virgules :



➤ Regroupement des propriétés

Certaines propriétés génériques prévoient une version raccourcie, permettant l'application de plusieurs valeurs en une seule déclaration. Ainsi, la propriété **font** rassemble les valeurs des propriétés **font-style**, **font-size**, **font-family**, **font-weight** et **line-height**. C'est ainsi que l'on pourra réduire la règle :



D'autres propriétés raccourcies pourront s'avérer bien utiles :

- **border** pour **border-width**, **border-style**, **border-color**.
- **background** pour **background-image**, **background-color**, **background-position**, **background-repeat**, **background-attachment**.

➤ Les sélecteurs et l'arborescence

Certaines des constructions syntaxiques spécifiques facilitent la sélection des éléments en CSS.

Selon la structure du document et principalement son arborescence (c'est-à-dire la hiérarchie des blocs, chacun étant placé « sous » l'éventuel bloc qui le contient), il est possible de pointer directement certains éléments en fonction de leur situation dans le document. Cette technique porte le nom de « sélection hiérarchique ».

L'arborescence du document permet de sélectionner plus aisément les divers éléments que l'on souhaite modifier. On peut comme cela se limiter aux balises dotées de certaines classes ou de certains identificateurs. Ainsi :

a.toto {background-color: yellow;}

ne désignera que les liens <a> de classe toto.

L'imbrication dans les sélecteurs permet aux règles CSS de ne concerner que les éléments descendant d'autres éléments précis. On peut ainsi ne cibler que les paragraphes contenus dans un bloc <div> sans influencer les autres paragraphes du document. Pour cela, on écrira le sélecteur sous la forme « ancêtre descendant » en séparant ces deux éléments d'un blanc.

Ainsi, la règle **a img {border-width: 0;}** ne supprimera que les bordures des images contenues dans un lien. Il est impératif que la balise descende de la balise <a> pour que la règle s'applique. Il n'est toutefois pas nécessaire que ces deux balises descendent directement l'une de l'autre : un ancêtre plus lointain conviendra.

➤ Autres sélecteurs hiérarchiques

Les normes CSS 2 et CSS 3 (actuellement en préparation) prévoient d'autres formes de sélection hiérarchique. On peut ainsi ne désigner que le premier enfant d'un élément, ne pointer que des éléments directement adjacents à un autre, etc. Il est même possible de prendre en compte les attributs des éléments pour les sélectionner. Ainsi, la règle **a[title="menu"] {color: blue;}** ne colorera en bleu que les liens possédant un attribut title de valeur menu. C'est par exemple le cas de lien en bleu. Pour des raisons de compatibilité, nous n'évoquerons pas ces procédés, malheureusement pas pris en charge par l'incontournable Internet Explorer 6. Ces techniques de sélection hiérarchique ayant cependant le vent en poupe, il est recommandé de s'y intéresser.

NOUVEAUTÉ Internet Explorer 7 change la donne

La dernière version du navigateur de Microsoft prend en charge plusieurs nouveaux sélecteurs CSS2 :

- **:first-child** désigne le premier élément de son élément parent.
- **div>p** : le signe « > » désigne un sélecteur d'enfant. Il s'applique à l'élément <p> enfant de <div>.
- **div+p** : le signe « + » désigne un sélecteur adjacent. Il s'applique à <p> lorsqu'il est précédé d'un élément <div>.
- **a[title="menu"]** : le sélecteur d'attribut s'applique sur un élément (ici <a>) lorsque l'attribut de l'élément (ici title) contient une valeur définie (ici "menu").

CHAP 5 : PAUSE-TP N°2

1. Gestion des couleurs

En abordant la question de la couleur, nous entrons de plein-pied dans l'outil CSS et le webdesign. Le mot « design » se rapporte au travail sur l'esthétique d'un objet utilitaire produit par l'industrie. Il a désormais conquis tous les secteurs d'activité et de production, notamment l'architecture, l'industrie, le graphisme (publicité, Web), etc. Il s'agit de rendre « beau » un objet conçu avant tout pour être utile. Le webdesign, ou « design de sites web », s'inscrit dans cette lignée, et désigne avant tout l'alliance entre le graphisme et le côté fonctionnel d'un site... deux notions pas forcément antagonistes. N'oublions pas que la beauté et le graphisme ne forment qu'une partie d'un tout : un site web non ergonomique ne subsistera pas bien longtemps. Il faut toujours s'efforcer à doser subtilement les côtés fonctionnel et esthétique.

Une charte graphique réussie comprend le choix judicieux d'une palette de couleurs conforme au thème général du site, à des critères de jugement subjectifs, ainsi que des paramètres plus classiques comme l'ergonomie générale et la lisibilité du document. L'importance des teintes est confirmée par l'histoire et la sociologie : toutes les cultures et tous les peuples ont créé leur symbolique des couleurs et associé à chacune sa valeur et son interprétation.

➤ Indiquer la couleur de texte

Première chose à savoir : la propriété qui permet de gérer la couleur du texte est **color**. Comment indiquer donc le nom de la couleur ?

- ✓ La méthode la plus simple et la plus pratique pour choisir une couleur est de taper son nom (**en anglais bien sûr**).

Le seul défaut de cette méthode est qu'il n'existe que 16 couleurs dites "standard" :

Couleur	Aperçu
white	
silver	
gray	
black	
red	
maroon	
lime	
green	
yellow	
olive	
blue	
navy	
fuchsia	
purple	
aqua	
teal	

```

Body
{
    Color :navy ;
}

```

Cela mettra tout le texte de la page en bleu marine

- ✓ Une autre méthode c'est **la notation hexadécimale**. Un nom de couleur en hexadécimal, ça ressemble à ça : #FF5A28. Pour faire simple, c'est une combinaison de lettres et de chiffres qui indiquent une couleur. On doit toujours commencer par écrire un dièse (#), suivi de 6 lettres ou chiffres allant de 0 à 9 et de A à F. Ces lettres ou chiffres fonctionnent deux par deux. Les 2 premiers indiquent une quantité de rouge, les 2 suivants une quantité de vert, et les 2 derniers une quantité de bleu. En mélangeant ces quantités (qui sont les composantes Rouge-Vert-Bleu de la couleur) on peut obtenir la couleur qu'on veut.

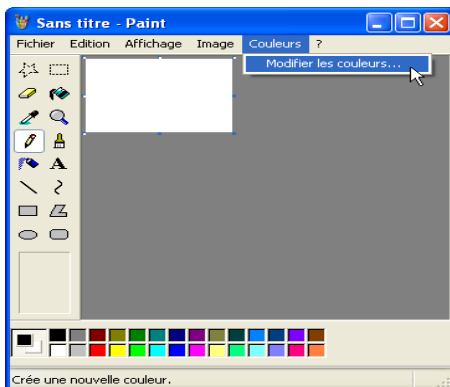
Ex : #000000 (noir) , #ffffff (blanc), #ff0000 (rouge), #008000 (vert), #0000ff (bleu), #ffff00 (jaune)

NB : Quand le codage hexadécimal d'une couleur se compose de trois paires jumelles, comme #ffffff, #cc55aa, #ffaa99, la syntaxe CSS permet de le condenser. On ne reporte pour cela qu'un seul caractère par couple, pour obtenir une notation à trois chiffres. Ainsi, #000000, #ffffff, #cc55aa et #ffaa99 deviennent respectivement #000, #fff, #c5a et #fa9. Cette technique ne pourra toutefois condenser des notations telles que #ffaa96, #3aaaaa, #00000f, car leurs paires de caractères consécutifs ne sont pas composées des mêmes chiffres hexadécimaux.

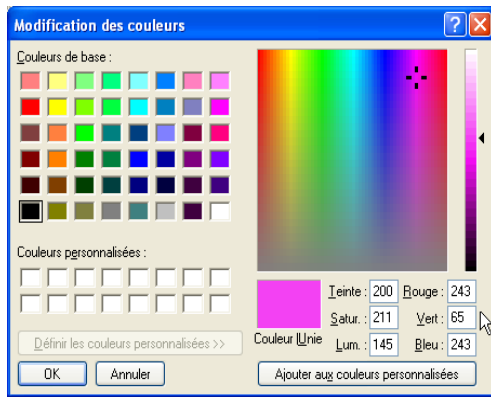
- ✓ Enfin, la notation RGB (Red-Green-Blue, en français Rouge-Vert-Bleu). Comme pour la notation hexadécimale, on doit définir une quantité de rouge, de vert et de bleu pour choisir une couleur.

Avec un logiciel tel que paint, vous pouvez trouver la couleur que vous désirez. Voici la marche à suivre :

1. Lancez le logiciel Paint depuis le menu Démarrer.
2. Rendez-vous dans le menu Couleurs / Modifier les couleurs :



3. Une fenêtre s'ouvre. Cliquez sur le bouton "Définir les couleurs personnalisées" en bas. Dans la zone qui apparaît à droite, faites bouger les curseurs pour sélectionner la couleur qui vous intéresse.
4. Supposons que je sois pris d'une envie folle d'écrire mes titres <h1> en rose barbie (supposons seulement). Je sélectionne la couleur dans la fenêtre, comme ceci :



5. On relève les quantités de Rouge-Vert-Bleu correspondantes indiquées en bas à droite de la fenêtre (ici 243-65-243). Je recopie ces valeurs dans cet ordre dans le fichier CSS, comme ceci :

```
h1
{
  text-align: center;
  color: rgb(243,65,243);
}
```

➤ Indiquer la couleur de fond

Contrairement à ce qu'on pourrait croire, le fond ne désigne pas forcément le fond de toute une page web. On peut aussi appliquer un fond uniquement aux titres, ou aux paragraphes, ou encore à certains mots d'un paragraphe.

Il faut tout d'abord savoir qu'il existe 2 types de fonds :

- Les fonds comportant une couleur
- Les fonds comportant une image de fond

Nous allons commencer à nous intéresser à la couleur de fond dans un premier temps, puis nous verrons dans **point 3(Bordure, arrière-plan et image)de cette PAUSE-TP** comment faire pour avoir une image de fond.

Pour indiquer une couleur de fond, on utilise la propriété CSS **background-color**. Elle s'utilise de la même manière que la propriété **color**, c'est-à-dire que vous pouvez taper le nom d'une couleur, l'écrire en notation hexadécimale ou encore utiliser la méthode RGB.

Pour indiquer la couleur de fond de la page web, il faut travailler sur la balise <body>

➤ Récapitulatif

✓ La symbolique des couleurs

Les signes et influences attribués aux couleurs par les cultures et les sociétés humaines sont variés mais pas toujours contradictoires. Ainsi le noir, symbole de deuil et de tristesse en Occident, n'a pas la même fonction dans les pays où ce rôle est dévolu à d'autres couleurs (notamment au blanc). Malgré tout, on peut brosser un tableau général de la symbolique des couleurs.

Couleur	Symbolique	Exemples d'utilisation
Blanc	Le vide, l'espace, la pureté, la propreté et la sobriété. C'est une couleur reposante et non agressive.	Intervient rarement explicitement ; souvent relégué au second plan (une page web aérée se devant de présenter un arrière-plan allégé). Il permet aussi de faire ressortir les éléments importants de la page.
Noir	Symbole de deuil ou de tristesse, le noir est également associé à l'art, à l'apparat et au luxe.	Un fond noir donne une impression d'élégance : on retrouve ainsi cette technique sur les sites web d'art, les boutiques de luxe ou les casinos. Le noir se marie très bien avec les autres teintes.
Gris	Malgré sa connotation de mélange, le gris est également une couleur passe-partout.	Comme le bleu, cette couleur est souvent associée aux sites technologiques ou informatiques : les ordinateurs et le matériel informatique sont généralement gris.
Vert	Représente la nature et tout ce qui s'y rapporte. C'est aussi une couleur qui confère une impression de fraîcheur saine : on la retrouve ainsi dans les sirops et chewing-gums rafraîchissants.	Sites en rapport avec la nature, la chasse ou les loisirs.
Bleu	Couleur froide par excellence, le bleu inspire la rigueur et la science. Il représente également de grands espaces comme le ciel ou la mer, conférant ainsi une impression de tranquillité.	Sites web technologiques ou scientifiques.
Jaune	Couleur stimulante évoquant le dynamisme, le jaune attire l'oeil... mais trop de jaune agresse.	Sites associés à des thèmes dynamiques (sport, loisir, publicité, médias).
Rouge	Le rouge (avec l'orange) est la couleur chaude par excellence. Couleur associée à plusieurs éléments très forts, tels que le feu, l'amour, le sang.	Sa force rehausse la pâleur générale et dirige le regard du visiteur. Un site à dominante rouge dégage chaleur et passion.

✓ Les notations de couleurs

Pour clarifier et synthétiser la situation, rien de tel qu'un tableau exprimant les 16 couleurs les plus fréquentes dans chacun des trois systèmes vus ci-dessus (les codes en hexadécimal court ne sont donnés que quand ils existent).

Couleur	Mot-clé	RGB	Hexadécimal	Hexadécimal court
Noir	black	rgb(0,0,0)	#000000	#000
Blanc	white	rgb(255,255,255)	#ffffff	#fff
Gris	gray	rgb(128,128,128)	#808080	
Argent	silver	rgb(192,192,192)	#c0c0c0	
Bleu	blue	rgb(0,0,255)	#0000ff	#00f
Bleu marine	navy	rgb(0,0,128)	#000080	
Cyan	cyan	rgb(0,255,255)	#00ffff	#0ff
Cyan foncé	teal	rgb(0,128,128)	#008080	

Vert	green	rgb(0,128,0)	#008000	
Vert olive	olive	rgb(128,128,0)	#808000	
Vert clair	lime	rgb(0,255,0)	#00ff00	#0f0
Lilas	fuchsia	rgb(255,0,255)	#ff00ff	#f0f
Violet	purple	rgb(128,0,128)	#800080	
Rouge	red	rgb(255,0,0)	#ff0000	#f00
Marron	maroon	rgb(128,0,0)	#800000	
Jaune	yellow	rgb(255,255,0)	#ffff00	#ff0

2. La typographie et la mise en forme de caractères

On l'oublie parfois, mais le fondement du Web est de présenter du contenu. Quelle que soit sa charte graphique, un site web sans contenu manquera son objectif et n'aura pas de visiteurs réguliers. Les styles CSS joignent l'utile à l'agréable et mettront en forme les portions de texte de vos documents.

➤ Les polices de caractères

Vous serez peut-être surpris d'apprendre que toute police de caractères retenue pour un site web ne sera pas systématiquement reproduite de la même manière sur les écrans des visiteurs – certains n'en tenant même aucun compte. En effet, les polices n'y sont qu'évoquées ; elles doivent être fournies par les ordinateurs de visualisation. Si la police précisée par le site web est absente du système client, elle y sera remplacée par une police par défaut : Times New Roman sur compatibles PC ; Times sur Mac. Cette dernière, peu adaptée aux textes de petite taille, gênera la lecture du site. Pour éviter ces mauvaises surprises et leurs inconvénients, il est recommandé de se limiter aux polices standards.

✓ Les polices standards

il est possible d'établir une liste de 11 polices de caractères dites standards, auxquelles tout webdesigner tâchera de se limiter : **Arial, Arial Black, Comic Sans MS, Courier New, Georgia, Impact Monotype, Symbol Monotype, Times New Roman, Trebuchet MS, Verdana, Webdings.**

✓ Déclarer la police en CSS

C'est la propriété font-family qui permet de contrôler la police affectée à un élément du site (qu'il s'agisse du document complet ou de l'une de ses parties). Le code suivant applique ainsi la police Trebuchet MS à l'ensemble du document :

```
body {
  font-family: 'Trebuchet MS';
}
```

Si Trebuchet MS est absente de l'ordinateur, elle y sera remplacée par Times. Pour éviter cela, on précisera plusieurs valeurs à font-family, ce qui en étendra les possibilités. Par exemple :

```
body {
  font-family: 'Trebuchet MS', times, verdana;
}
```

➤ Les unités de taille de polices

Deux systèmes permettent d'indiquer les dimensions des éléments en CSS (notamment les tailles des polices) : les unités de taille fixe et les unités de taille relative. C'est la propriété font-size qui détermine la taille de la police d'un élément.

```
p {  
  font-size: 14px;  
}
```

✓ Les unités de taille fixe

Les unités de taille fixe (ou unités absolues) sont le point (1 **pt** vaut environ 0,35 mm), le pica (1 **pc** vaut environ 4,22 mm), le centimètre (**cm**), le millimètre (**mm**) et le pouce (1 in vaut environ 2,54 cm). Le W3C conseille de limiter leur usage à des médias de sortie connus, aux propriétés déterminées. En clair, on les évitera sur écran d'ordinateur, chaque moniteur étant particulier (de par sa taille de diagonale, sa résolution, son nombre de couleurs, etc.). De telles unités sont toutefois parfaitement adéquates à une sortie sur papier. Évitez vous aussi d'utiliser ces unités fixes dans l'élaboration de votre site web et pour le calcul de vos dimensions et tailles de polices.

✓ Les unités de taille relative et pourcentages

Les unités relatives sont le cadratin (**em**), la hauteur d'« x » (**ex**), le pourcentage (%) et le pixel (**px**). 1 em représente la taille d'un caractère (ainsi que l'espace pour ses jambages) dans la police de référence. 1 ex correspond à la hauteur du caractère minuscule « x », sans jambages, dans la police de référence. Le pourcentage, s'il se prête bien aux textes et polices de caractères, ne leur est pas spécifique. Il se définit relativement à la taille de référence dans le conteneur de l'élément.

Pour les trois premiers unités, la taille de la police de référence se transmet par héritage : dans le cas d'éléments imbriqués, la police de référence change à chaque nouveau conteneur. Ainsi, définir une taille de référence de 2 em (ou 80%) dans un paragraphe puis une taille de 2 em (ou 80%) dans un de ses éléments enfants, on attribue à ce dernier une taille de 2 em(ou 80%) par rapport à 2 em(ou 80%), soit 4 em(ou 64%) !

Pour conclure cette liste, le pixel (dit « unité relative » car sa taille dépend de la plateforme et de l'écran d'ordinateur) est sans doute l'unité la plus utilisée. Sur un système donné, il restera une unité fixe partout dans le document.

✓ Les mots-clés de tailles

Comme pour les couleurs, CSS propose des mots-clés pour définir les tailles de polices. Malheureusement, il n'en existe que sept. Par ordre croissant, ce sont **xxsmall**, **x-small**, **small**, **medium**, **large**, **x-large**, et **xx-large**.

```
h1 {  
  font-size: large;  
}
```

Ces tailles sont laissées à l'appréciation de chaque navigateur. Internet Explorer et Netscape Navigator afficheront ainsi des tailles différentes pour un texte medium. Ce spectre limité et ces différences d'appréciation limitent grandement l'intérêt de cette technique pour le le choix des tailles de polices.

➤ Styles et effets sur les caractères

CSS propose de nombreux styles typographiques. Nous distinguerons les styles prévus pour les caractères (**couleur** (que nous avons déjà vue avec la propriété **color**), **italique**, **la graisse**, **soulignement**, etc.) et ceux portant sur les mots et les paragraphes (**interlignage**, **justification**, **crénage**, etc.).

✓ La graisse, les italiques et les obliques

La propriété **font-weight** définit la graisse de caractères. Elle accepte les valeurs :

- **normal** : graisse normale.
- **bold** : gras.
- **lighter** : moins gras que la référence.
- **bolder** : plus gras que la référence.
- **100, 200, ... 900** : chaque valeur définit un niveau de graisse différent. Ces valeurs ne sont malheureusement pas souvent prises en compte par les différents navigateurs. La graisse d'un caractère dépend directement du type de police utilisé. D'autre part, **une grande partie des polices ne reconnaîtront que les valeurs normal et bold.**

```
.gras {  
  font-weight: bold;  
}
```

La propriété **font-style** gère les italiques et les obliques. Elle admet les valeurs :

- **normal** : police droite.
- **italic** : spécifie une police dite « italique » dans la base de données de polices du navigateur (c'est le cas de toute police dont le nom comporte l'un des mots « Italic », « Cursive », ou « Kursiv »). À défaut, se rabat sur une police étiquetée « oblique ».
- **oblique** : **spécifie une police dite « oblique » dans la base de données de polices du navigateur** (c'est le cas de toute police dont le nom comporte d'un des mots « Oblique », « Slanted », ou « Incline »). Rares sont les polices de caractères disposant d'une variante « oblique ».

```
.italique {  
  font-style: italic;  
}
```

✓ Caractères soulignés, surlignés, barrés, clignotants

La propriété **text-decoration** permet :

- de souligner le texte (valeur **underline**) ;
- de surligner le texte (valeur **overline**) ;
- de barrer le texte (valeur **line-through**) ;
- de faire clignoter le texte (valeur **blink**).

Internet Explorer ne reconnaît pas le clignotement (blink) – on y recourra avec modération, car cette propriété agresse ou gêne souvent les utilisateurs.

NORME FACULTATIVE : blink n'est pas imposé

Le W3C n'oblige pas les navigateurs à prendre en charge la fonctionnalité de clignotement.

On peut cumuler toutes ces fonctionnalités de décoration du texte, en précisant plusieurs valeurs à la propriété **text-decoration**. Le code suivant produira ainsi un titre souligné et surligné :


```
h1 {  
text-decoration: underline overline;  
}
```

On recourt très fréquemment à la propriété `text-decoration` pour modifier le soulignement des liens hypertextes. Ainsi, pour mettre en place des liens soulignés uniquement lors de leur survol par le pointeur de la souris, on pourra écrire :

```
a {  
text-decoration: none;  
}  
a:hover {  
text-decoration: underline;  
}
```

✓ La casse : minuscule et majuscule

La casse est la prise en compte des majuscules et des minuscules des caractères. Elle se traduit en CSS par la propriété héritée **`text-transform`**, qui admet quatre valeurs :

- **`capitalize`** : seule la première lettre de chaque mot du texte sera affichée en majuscule.
- **`lowercase`** : tout le texte sera affiché en minuscules.
- **`uppercase`** : tout le texte sera affiché en majuscules.
- **`none`** : le texte ne sera pas modifié.

```
.majuscules {  
text-transform: uppercase;  
}
```

```
<p class="majuscules">Texte en majuscule</p>
```

➤ Les styles et effets sur les mots et paragraphes

✓ Interlignage de texte

L'**interligne** est l'**espace séparant deux lignes consécutives d'un paragraphe** (à ne pas confondre avec les marges de paragraphe, qui ne portent que sur son périmètre). Les paragraphes et autres éléments contenant du texte ont une valeur d'interligne par défaut qui dépend des navigateurs. Elle est d'environ 1,2 em, soit un peu plus que la hauteur d'un caractère. La propriété CSS mettant en place l'interligne s'appelle `line-height`. Elle admet pour valeurs un nombre, un pourcentage, ou le mot-clé normal. Toutes les unités sont acceptées ; « em » est toutefois conseillée.

```
.interligne {  
line-height: 2em;  
}
```

```
<p class="interligne">Test de paragraphe pour montrer la hauteur de  
l'interligne fixé à 2 em, soit le double de la hauteur de caractère. </p>
```

✓ Le crénage

Le **crénage** (ou **interlettrage**) est la distance séparant deux caractères consécutifs. Toutes les balises disposent d'une valeur de crénage par défaut, qu'on peut modifier grâce à la propriété CSS **letter-spacing** (une valeur négative ayant pour effet de resserrer les caractères).

```
.ecart {
  letter-spacing: 0.4em;
}
.rapproch {
  letter-spacing: -3px;
}
```

✓ L'espace entre les mots

La propriété **word-spacing** règle l'espace séparant deux mots consécutifs. Elle se comporte comme **letter-spacing** et admet les mêmes valeurs, mais ne porte que sur les mots entiers. Cette fonctionnalité n'est proposée par Internet Explorer qu'à partir de sa version 6. Par exemple :

```
.ecart {
  word-spacing: 0.4em;
}
.rapproch {
  word-spacing: -3px;
}
```

✓ Définir la justification du texte

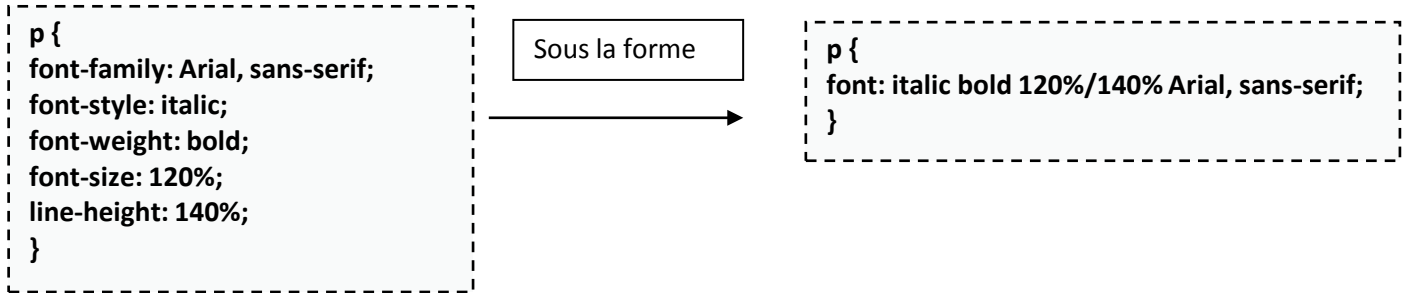
Trois comportements vis-à-vis des marges sont possibles pour chaque paragraphe : alignement à gauche, alignement à droite, justification. Cette dernière possibilité, souvent utilisée dans les livres, se traduit par un alignement simultané sur les deux marges, en jouant automatiquement sur l'interlettrage et l'espace entre les mots. La propriété CSS définissant l'alignement de texte est **text-align**. Elle admet les valeurs **left** (alignement à gauche), **right** (alignement à droite), **center** (texte centré), **justify** (texte justifié) et **normal** (comportement par défaut).

```
.interligne {
  text-align: justify;
}
```

```
<p class="interligne">Test de paragraphe démontrant l'alignement
justifié, c'est-à-dire collé à gauche et à droite du bloc.</p>
```

➤ La notation raccourcie

Pour éviter l'accumulation de déclarations de polices, le W3C a prévu une notation raccourcie des propriétés débutant par « font. ». Comme nous l'avons déjà évoqué dans le chapitre d'introduction aux feuilles de styles, il est possible de synthétiser la règle suivante :



3. Bordure, arrière plan et image

Dans un document web graphique, la prise en compte de l'environnement des éléments (bordures, arrière-plans et images de fond) constitue la dernière étape majeure avant de passer aux positionnements d'objets en CSS à proprement parler.

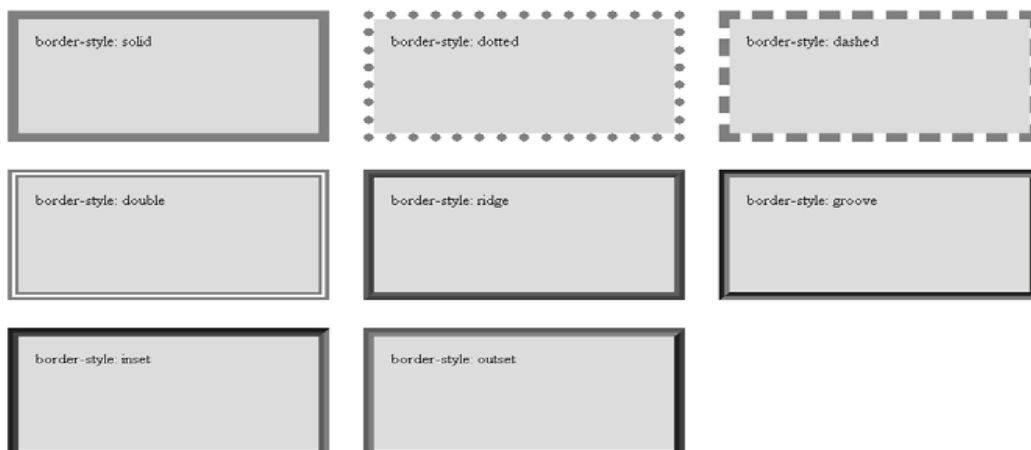
➤ Mettre en forme les bordures

CSS permet d'entourer très simplement les divers **éléments de type bloc** d'une page web par des bordures aux caractéristiques (style, épaisseur, couleurs) libres. Cette possibilité est également offerte **aux éléments en ligne remplacés** (``, `<input />`, `<textarea></textarea>`, `<select></select>` et `<object></object>`), qui possèdent des dimensions par défaut (hauteur et largeur).

✓ Les différents styles de bordures

La propriété **border-style** met en place le type des bordures d'un élément. Tous les navigateurs ne reconnaissent pas les dix styles de bordures possibles :

- **dashed** : tirets ;
- **dotted** : pointillés ;
- **double** : deux traits pleins d'épaisseur égale et séparés par un espace vide de même épaisseur ;
- **groove** : effet 3D gravé dans la page, opposé de ridge ;
- **hidden** : pas de bordure mais influe sur la bordure mitoyenne ;
- **inset** : effet entrant, élément incrusté dans la page (opposé d'outset) ;
- **none** : pas de bordure ; équivaut à `border-width: 0` ;
- **outset** : effet sortant, élément extrudé de la page (opposé d'inset) ;
- **ridge** : effet 3D sortant de la page, opposé de groove ;
- **solid** : trait plein.



Chacun des quatre côtés d'un élément peut aussi être représenté différemment. Pour cela, on précise à la suite plusieurs valeurs de **border-style**, dont l'interprétation dépendra de leur nombre.

- deux valeurs seront respectivement affectées aux côtés horizontaux et verticaux du cadre. Par exemple : **border-style: solid dotted;**
- trois valeurs concerneront tour à tour le haut, les côtés verticaux, et le bas. Par exemple : **border-style: solid double dotted;**
- quatre valeurs décriront les styles des quatre côtés en tournant dans le sens horaire haut, droit, bas, et gauche. Par exemple : **border-style: solid double dotted ridge;**

Il existe une autre possibilité : les propriétés **border-top-style (haut)**, **border-right-style (droit)**, **border-bottom-style (bas)** et **border-left-style (gauche)** s'appliqueront directement au style de bordure d'un côté.

BUG Internet Explorer

Internet Explorer 6 présente un bug de rendu visuel : les bordures d'un pixel de large dont le style est défini en dotted (pointillés) sont en réalité affichées en style dashed (tirets) ! La version 7 du navigateur corrige ce bug.

✓ L'épaisseur des bordures

La propriété **border-width** définit l'épaisseur des bordures, et n'a de sens qu'en accompagnement d'un style (**border-style**) ou d'une couleur de bordure (**border-color**). Certains navigateurs n'interprètent d'ailleurs que les bordures ayant renseigné ces deux propriétés. On peut préciser les épaisseurs de plusieurs manières (l'interprétation des trois premières dépend du navigateur) :

- **thin** : bordure fine ;
- **medium** : bordure moyenne ;
- **thick** : bordure épaisse ;
- avec une mention numérique de longueur reprenant la syntaxe habituelle.

La syntaxe des mentions de longueurs est exposée plus haut, dans la section portant sur les unités de taille de polices. À nouveau, fournir plusieurs valeurs permettra de représenter différemment les quatre côtés de l'élément. Comme pour le style, l'affectation des valeurs dépend de la taille de leur liste :

- Deux valeurs portent sur les côtés horizontaux puis verticaux. Par exemple : **border-width: 6px 2px;**
- Trois valeurs représentent le haut, les côtés latéraux, et le bas. Par exemple : **border-width: thin medium 3em;**
- Quatre valeurs définissent les côtés en partant du haut et en tournant dans le sens horaire. Par exemple : **border-width: thin medium 3em 1em;**

On peut ici encore définir directement l'épaisseur de chaque côté avec les propriétés **border-top-width**, **border-right-width**, **border-bottom-width** et **border-left-width**.

✓ La couleur des bordures

Elle est mise en place par la propriété **border-color**, qui ne s'applique qu'en complément d'un style (**border-style**) ou d'une épaisseur de bordure (**border-width**). Sans cela, la bordure est inexistante.

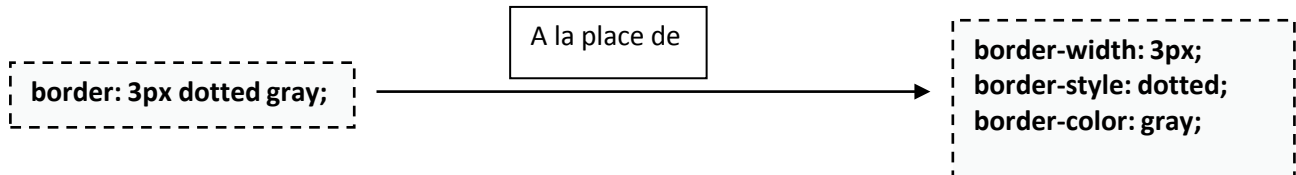
Encore une fois, l'ordre d'attribution des valeurs de couleurs se fait dans le sens horaire en partant du haut, en les complétant éventuellement par symétrie.

- Deux valeurs définissent ainsi une couleur par orientation (horizontale ou verticale) de côté. Exemple : **border-color: #fc0 #ccc;**
- Trois valeurs permettent de plus de distinguer le haut du bas. Exemple : **border-color: blue green red;**
- Quatre valeurs individualisent chaque côté. Exemple : **border-color: blue green red yellow;**

De nouveau, les propriétés **border-top-color**, **border-right-color**, **border-bottom-color** et **border-left-color** s'appliquent directement à un côté précis.

✓ Notation raccourcie

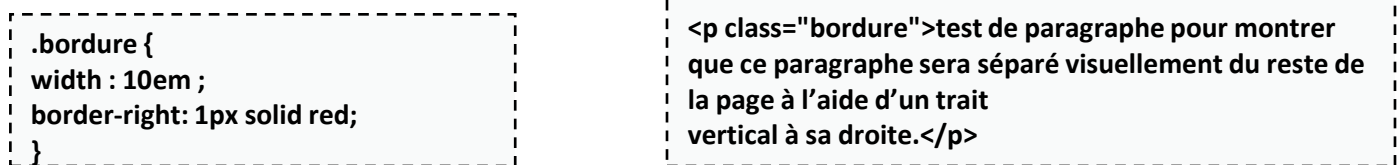
La propriété générale **border** rassemble tous ces aspects en une seule instruction. Pour obtenir une bordure en pointillés gris large de trois pixels, il suffit ainsi d'écrire :



✓ Afficher un trait vertical

Par une utilisation astucieuse des bordures, on peut encore mettre en place une ligne verticale dans un document. L'élément **<hr />** (horizontal rule) est un séparateur qui trace un trait horizontal. L'absence d'élément décrivant une ligne verticale incite souvent les designers à recourir à diverses « bidouilles » pour les créer. Les plus courantes impliquent l'ajout de colonnes de tableau ou de blocs **<div>** verticaux et positionnés.

Dans la plupart des cas, ces complications sont inutiles, et une simple bordure à droite ou à gauche de l'élément fait l'affaire. Pour créer un paragraphe de texte clairement séparé du reste du contenu par un trait vertical, on pourra ainsi écrire :



Cette méthode convient aussi pour marquer une séparation entre un menu et le contenu de la page.

➤ Arrière-plans et images de fond

Les propriétés CSS relatives aux couleurs et images d'arrière-plan permettent de prévoir une mise en page indépendante du document : une simple modification de la feuille de styles suffira à transformer l'ensemble du site.

✓ Insérer une image d'arrière-plan

La propriété **background-image** associe une image de fond à l'élément sur lequel elle porte. Pour mettre en place une image d'arrière-plan valable pour tout le document, on la placera sous la balise **<body>**.

Par défaut, cette image sera répétée en damier (ou papier peint) à partir du coin supérieur gauche. On peut toutefois modifier ce comportement en jouant sur les autres attributs CSS de la propriété **background**. À nouveau, l'absence d'un mécanisme d'héritage a peu d'impact : le comportement par défaut étant l'absence de tout fond, les éléments imbriqués apparaîtront comme ceux de leurs parents. Deux valeurs sont possibles : **none** et **url(chemin-vers-l.image)**. Voici des exemples :

```
div#global {background-image: url (dossier/fond.jpg);}
div#global {background-image: url(http://www.monsite.com/dossier/fond.jpg);}
```

Soulignons l'absence inhabituelle d'apostrophes simples ou doubles autour des noms des images.

ASTUCE : Contourner un bogue de Netscape Navigator

Les versions 4 et 6 de Netscape Navigator peuvent poser des problèmes d'affichage lors de l'imbrication de tableaux, et répéter un arrière-plan dans les cellules des tableaux imbriqués.

On corrige ce bogue en spécifiant un arrière-plan sans image dans les éventuelles cellules concernées.

✓ Positionner l'image à sa convenance

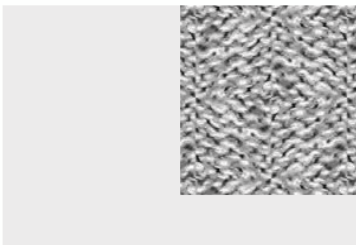
Avec **background-position**, on pourra placer dans un élément une image d'arrièreplan définie par **background-image**. Cet attribut s'utilise surtout en l'absence de répétition (**background-repeat**). La première valeur (exprimée de manière absolue par une longueur ou relative par un pourcentage) spécifie la position horizontale par rapport au bord gauche. L'éventuelle deuxième valeur (démarquée de la première par un blanc) portera sur la position verticale (par rapport au bord supérieur). En son absence, c'est la valeur par défaut center (ou 50%) qui prendra effet. Voici quelques règles :

- les nombres négatifs sont autorisés ;
- il est interdit de mêler valeurs relatives et absolues (dans le cas des positions) ;
- les valeurs relatives sont des pourcentages calculés par rapport aux dimensions de l'image.

En somme, ces deux valeurs placent le coin supérieur gauche de l'image dans l'espace. Pour mémoire les correspondances sont :

- 0% 0% = left top
- 50% 50% = center center
- 100% 100% = right bottom

Par exemple :



```
div#global {  
width: 300px;  
height: 200px;  
background-color: yellow;  
background-image: url(dossier/fond.jpg);  
background-repeat: no-repeat ;  
background-position: right top;  
}
```

✓ Répéter l'image ou non

Par défaut, toute image d'arrière-plan se répète comme une mosaïque en emplissant l'espace de l'élément qui la contient. La propriété **background-repeat** permet de modifier ce comportement et reconnaît pour cela quatre valeurs :

- **repeat** : c'est la valeur et le comportement par défaut, où l'image se répète sur les deux axes.
- **no-repeat** : l'image n'apparaît qu'une fois, sans répétition.
- **repeat-x** : l'image ne se répète que dans le sens horizontal (exemple : frise).
- **repeat-y** : l'image ne se répète que dans le sens vertical (exemple : bordure graphique verticale ou marge).

```
body {  
background-image: url(frise.jpg);  
background-repeat: repeat-x;  
background-position: top;  
}
```

✓ Fixer l'image par rapport au contenu

Une image d'arrière-plan accompagne par défaut son élément si celui-ci est déplacé dans une barre d'ascenseur (scroll). On peut toutefois la fixer par rapport à la fenêtre du navigateur avec la propriété **background-attachment**. Pour mieux comprendre le principe, il suffit d'appliquer une image de fond au document entier :

```
body {
background-image: url(image.jpg);
background-repeat: no-repeat;
background-position: center right;
}
```

Placez à présent dans ce document un très long texte pour servir d'exemple. Le texte d'exemple ne tenant pas sur la fenêtre, une barre d'ascenseur apparaît sur la droite. Déplacer cet ascenseur provoque le comportement par défaut de l'image de fond : elle accompagne le mouvement jusqu'à sortir complètement du champ. Ajoutons à présent la déclaration suivante à l'élément **body** :

```
background-attachment: fixed;
```

L'image n'est plus attachée au contenu, mais à la fenêtre. Elle reste en place, quelle que soit la position de l'ascenseur. C'est donc une propriété très intéressante pour placer certains éléments de mise en page devant rester figés quel que soit le contenu du document (par exemple, des logos).

✓ Notation raccourcie

Une notation abrégée permet ici encore d'alléger les suites de déclarations semblables. La propriété **background** réunit toutes les caractéristiques d'arrière-plan décrites dans ce chapitre.

```
body {
background-color: white;
background-image: url(image.jpg);
background-repeat: no-repeat;
background-position: center right;
background-attachment: scroll;
}
```

se résumera
comme suit:

```
body {
background: white url(image.jpg) center
right no-repeat scroll;
}
```

CHAP 6 : PAUSE-TP N°3

1. Les listes à puces

➤ Différents types de listes (XHTML)

Tout d'abord, il faut savoir qu'il existe 3 types de listes à puces :

- Les listes non ordonnées
- Les listes ordonnées
- Les listes de définitions (plus rares)s

✓ Liste non ordonnée

Une liste non ordonnée ressemble à ceci :

- Fraises
- Framboises
- Cerises

C'est un système qui nous permet de faire une liste d'éléments, sans notion d'ordre (il n'y a pas de "premier" ni de "dernier"). Il suffit d'utiliser la balise `` que l'on referme un peu plus loin avec un ``. on va écrire chacun des éléments de la liste entre 2 balises ``. Toutes ces balises doivent se trouver entre `` et ``

```
<ul>
  <li>Fraises</li>
  <li>Framboises</li>
  <li>Cerises</li>
</ul>
```

Vous pouvez imbriquer des listes à puces (créer une liste à puce dans une liste à puces). Si vous voulez faire ça, ouvrez une seconde balise `` à l'intérieur d'un élément ``.

✓ Liste ordonnée

Pour faire une liste ordonnée, on ne va pas se fouler. On ne change que la balise `` qui devient cette fois ``. A l'intérieur, on ne change rien : on utilise toujours des balises ``. L'ordre dans lequel vous mettez les éléments de la liste est important. Le premier `` sera l'élément n°1, le second sera le n°2...

```
<ol>
  <li>Je me lève</li>
  <li>Je mange et je bois</li>
  <li>Je retourne me coucher</li>
</ol>
```

✓ Liste de définitions

On indique le début et la fin de la liste à l'aide d'une balise. Ici il ne s'agit pas de ``, ni de `` mais de `<dl>` (c'est l'abréviation de "Definition List").

On n'utilise plus la balise pour indiquer les différents éléments de la liste. En effet, dans le cas d'une liste de définitions, il y a 2 types d'éléments différents :

- Les mots
- Et leur définition

Les mots se trouvent entre <dt></dt>, et les définitions entre <dd></dd>.

Ce qu'il faut bien comprendre, c'est qu'on doit mettre d'abord le mot (<dt>), ensuite sa définition (<dd>). Si vous voulez mettre une seconde définition, il faut recommencer : un dt et ensuite un dd.

```
<dl>
  <dt>Chat</dt>
  <dd>Animal à 4 pattes qui fait "Miaou !"</dd>
  <dt>Chien</dt>
  <dd>Animal à 4 pattes qui fait "Ouaf Ouaf !"</dd>
  <dt>Prof de maths</dt>
  <dd>Personne qui enseigne des choses que personne ne comprend</dd>
</dl>
```

➤ Décorez vos listes (CSS)

✓ Retrait des listes

La première des propriétés que je veux vous montrer est très facile à utiliser. Elle permet d'indiquer si on veut que la liste soit mise en retrait ou non.

Cette propriété s'appelle **list-style-position**, et peut prendre 2 valeurs :

- **inside** : la liste n'est pas mise en retrait.
- **outside** : la liste est mise en retrait (par défaut).

```
.pas_de_retrait
{
  list-style-position: inside;
}

.retrait
{
  list-style-position:
  outside;
```

```
<p>Voici une liste avec retrait (par défaut) :</p>
<ul class="retrait">
  <li>Liste<br />à puces</li>
  <li>Ligne 1<br />Ligne 2</li>
  <li>Vous pouvez voir<br />que le texte est décalé sur la droite</li>
</ul>
<p>Voici une liste sans retrait :</p>
<ul class="pas_de_retrait">
  <li>Liste<br />à puces</li>
  <li>Ligne 1<br />Ligne 2</li>
  <li>Vous pouvez voir<br />que le texte n'est pas décalé sur la droite</li>
</ul>
```

Pour bien voir la différence au niveau du retrait, il faut que les éléments de la listes comportent au moins 2 lignes (c'est pour ça que j'ai mis un
). En pratique, on ne désactive pas le retrait des listes à puces, sauf quand on se sert des listes pour créer le menu de son site comme on le verra plus tard.

✓ Représentation de la puce

Nettement plus intéressante, la propriété **list-style-type** vous permet de changer l'apparence de la puce. La propriété **list-style-type** peut prendre de nombreuses valeurs. Certaines d'entre elles concernent uniquement les listes non ordonnées, d'autres concernent uniquement les listes ordonnées :

- Pour les listes non ordonnées () :
 - disc : un disque noir (par défaut).
 - circle : un cercle.
 - square : un carré.
 - none : aucune puce ne sera utilisée.
- Pour les listes ordonnées (), le choix est vaste :
 - decimal : des nombres 1, 2, 3, 4, 5... (par défaut)
 - decimal-leading-zero : des nombres commençant par zéro (01, 02, 03, 04, 05...).
 - upper-roman : numérotation romaine, en majuscules (I, II, III, IV, V...)
 - lower-roman : numérotation romaine, en minuscules (i, ii, iii, iv, v...)
 - upper-alpha : numérotation alphabétique, en majuscules (A, B, C, D, E...)
 - lower-alpha : numérotation alphabétique, en minuscules (a, b, c, d, e...)
 - lower-greek : numérotation grecque.

Il existe en réalité d'autres valeurs possibles pour les listes ordonnées, comme la numérotation arménienne, géorgienne, hébraïque, japonaise etc...

```
<h2>Quelques listes non ordonnées</h2>
<p>Voici une liste à puce non ordonnée sans modification (= <em>disc</em>) :</p>
<ul>
  <li>Liste</li>
  <li>à</li>
  <li>puces</li>
</ul>
<p>Avec <em>circle</em> :</p>
<ul class="cercle">
  <li>Liste</li>
  <li>à</li>
  <li>puces</li>
</ul>
<p>Avec <em>square</em> :</p>
<ul class="carre">
  <li>Liste</li>
  <li>à</li>
  <li>puces</li>
</ul>
<p>Avec <em>none</em> (sans puces) :</p>
<ul class="rien">
  <li>Liste</li>
  <li>à</li>
  <li>puces</li>
</ul>
```

```
<h2>Quelques listes ordonnées</h2>
<p>Liste à puces ordonnée sans modification (= <em>decimal</em>)</p>
<ol>
<li>Un</li>
<li>Deux</li>
<li>Trois</li>
<li>Quatre</li>
</ol>
<p>Avec <em>decimal-leading-zero</em> (<strong>Ne fonctionne pas sous IE</strong>) :</p>
<ol class="commence_a_zero">
<li>Un</li>
<li>Deux</li>
<li>Trois</li>
<li>Quatre</li>
</ol>
<p>Avec <em>lower-alpha</em> :</p>
<ol class="lettres_minuscules">
<li>Un</li>
<li>Deux</li>
<li>Trois</li>
<li>Quatre</li>
</ol>
<p>Avec <em>upper-roman</em> :</p>
<ol class="chiffres_romains">
<li>Un</li>
<li>Deux</li>
<li>Trois</li>
<li>Quatre</li>
</ol>
<p>Avec <em>lower-greek</em> (<strong>Ne fonctionne pas sous IE</strong>) :</p>
<ol class="lettres_grecques">
<li>Un</li>
<li>Deux</li>
<li>Trois</li>
<li>Quatre</li>
</ol>
```

```
/* Listes à puces non ordonnées */
```

```
.cercle
{
  list-style-type: circle;
}
.carre
{
  list-style-type: square;
}
.rien
{
  list-style-type: none;
}
```

```
.commence_a_zero
{
  list-style-type: decimal-leading-zero;
}
.lettres_minuscules
{
  list-style-type: lower-alpha;
}
.chiffres_romains
{
  list-style-type: upper-roman;
}
.lettres_grecques
{
  list-style-type: lower-greek;
}

/* Quelques styles supplémentaires juste pour la présentation
En plus ça vous fait quelques rappels ;o) */

h2
{
  text-indent: 20px;
  font-family: Arial, Verdana, "Times New Roman", serif;
}
em
{
  background-color: yellow;
}
strong
{
  color: red;
}
```

✓ Changer l'image de la puce

Vous pouvez créer votre propre représentation de style avec **list-style-image**. Vous devez lui indiquer comme valeur "url" suivi de l'adresse de l'image à utiliser. Ça fonctionne exactement comme la propriété background (qui permet d'indiquer une image de fond, si vous vous souvenez bien).

```
ul /* Ma liste aura des puces en forme de dossiers */
{
  list-style-image: url("dossier.png");
}

/* Juste pour améliorer la présentation : */
a
{
  color: blue;
  text-decoration: none;
}
```

```
}  
a:hover  
{  
  text-decoration: underline;  
}
```

2. Mise en boîte

➤ Block et Inline

comment reconnaître une balise inline d'une balise block ?

C'est en fait assez facile :

- **block** : une balise de type "block" sur votre page web crée automatiquement un retour à la ligne avant et après. Il suffit d'imaginer tout simplement un bloc. Votre page web sera en fait constituée d'une série de blocks à la suite les uns des autres. Mais vous verrez qu'en plus, il est possible de mettre un block à l'intérieur d'un autre, ce qui va augmenter considérablement nos possibilités pour créer le design de notre site !
- **inline** : une balise de type "inline" se trouve obligatoirement à l'intérieur d'une balise "block". Une balise inline ne crée pas de retour à la ligne, le texte qui se trouve à l'intérieur s'écrit donc à la suite du texte précédent, sur la même ligne (c'est pour cela que l'on parle de balise "en ligne").

➤ Les balises universelles

Ce sont des balises qui n'ont aucun sens particulier (contrairement à <p> qui veut dire "paragraphe", "important" etc...). Le principal intérêt de ces balises c'est de pouvoir appliquer une class (ou un id) pour le css quand aucune autre balise ne convient.

Il existe 2 balises génériques, et comme par hasard la seule différence entre les deux c'est que l'une d'elle est inline, l'autre est block :

- **** (inline)
- **<div></div>** (block) : c'est aussi une balise qui n'a aucun sens, comme span, sauf que celle-là est de type block. On va pas mal s'en servir dans les chapitres qui suivent, notamment pour créer le design de votre site.

➤ Respectez la sémantique !

Les balises universelles sont "pratiques" dans certains cas, certes, mais attention à ne pas en abuser. Je tiens à vous avertir de suite : beaucoup de webmasters mettent des <div> et des trop souvent, et oublient que d'autres balises plus adaptées existent.

Voici 2 exemples :

- Exemple d'un span inutile :
Je ne devrais jamais voir dans un de vos codes :

... alors qu'il existe la balise qui sert à ça !
- Exemple d'un div inutile :
De la même manière, ceci est complètement absurde :

```
<div class="titre">
```

... puisqu'il existe des balises faites spécialement pour les titres (<h1><h2><h3>...)

En XHTML, on vous demande d'utiliser tant que possible des balises qui ont un sens : **on appelle ça respecter la sémantique**. L'avantage c'est que si vous respectez ceci, votre code aura une certaine "logique". Lorsque le robot de Google viendra référencer votre site, il "comprendra" le sens des balises et cela pourra améliorer votre position dans le moteur de recherche.

➤ La taille

Un block a des dimensions précises. Il a une largeur et une hauteur. On dispose de 2 propriétés CSS :

- **width** : c'est la largeur du block. A exprimer en pixels (px) ou en pourcentage (%).
- **height** : c'est la hauteur du block. Là encore, on l'exprime soit en pixels (px), soit en pourcentage (%).

Les balises block prennent 100% de la largeur disponible par défaut. Mais on peut modifier la taille en css :

```
p{
  width: 50%;
  text-align: justify; /* Texte justifié pour mieux voir la largeur du block */
}
```

Les pourcentages seront utiles pour créer un design qui s'adapte automatiquement à la résolution du visiteur.

Toutefois, il se peut que vous ayez besoin de créer des blocks ayant une dimension précise en pixels .

Il vous sera aussi très pratique de demander à ce qu'un block ait une taille minimale ou maximale. C'est là qu'entrent en jeu les 4 propriétés CSS suivantes :

- **min-width** : largeur minimale
- **min-height** : hauteur minimale
- **max-width** : largeur maximale
- **max-height** : hauteur maximale

Attention cependant, ces propriétés ne fonctionnent pas sous IE 6 et fonctionnent partiellement sous IE 7. Evitez de les utiliser tant qu'IE 6 est encore assez répandu.

➤ "Couper" un block avec un overflow

Supposons que vous ayez un long paragraphe et que vous vouliez qu'il fasse 250px de large et 100px de haut.

```
p{
  width: 250px;
  height: 100px;
  text-align: justify;
}
```

Mais si le texte n'a pas assez de place, le block va s'agrandir de manière à ce que tout soit visible. Si vous voulez "couper" votre paragraphe pour qu'il soit d'une dimension exacte de 250x100, vous allez devoir utiliser la propriété CSS **overflow**. Les valeurs que peut prendre overflow sont les suivantes :

- **visible** (par défaut) : si le texte dépasse les limites de taille, le block est agrandi de manière à ce que tout soit visible. C'est ce que nous venons de constater.
- **hidden** : si le texte dépasse les limites, il sera tout simplement coupé. On ne pourra pas voir tout le texte.
- **scroll** : là encore, le texte sera coupé s'il dépasse les limites. Sauf que cette fois, le navigateur mettra en place des barres de défilement pour qu'on puisse voir tout le texte. C'est un peu comme un cadre à l'intérieur de la page.
- **auto** : mode "pilote automatique" :p. En gros, c'est le navigateur qui décide ou pas de mettre des barres de défilement (il en mettra si c'est nécessaire).

```
p /* Tous les paragraphes auront ces propriétés CSS */{
  width: 250px;
  height: 100px;
  text-align: justify;
}/* Et chacun des paragraphes aura en plus une valeur d'overflow différente */
.coupe{
  overflow: hidden;
}
.barres_defilement{
  overflow: scroll;
}
.auto{
  overflow: auto;
}
```

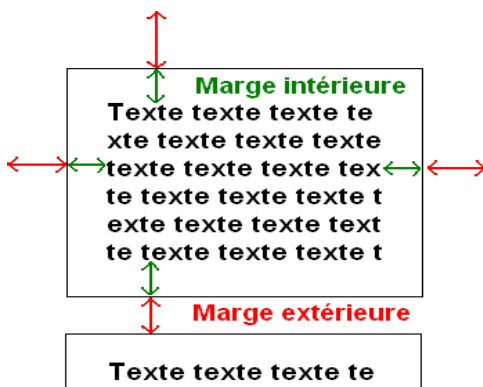
➤ Les marges

C'est vital. Si vous ne savez pas précisément comment fonctionnent les marges des blocks, vous serez bien embêtés lors de la création de votre design !

Tous les blocks possèdent des marges. Ce qui est important, c'est de savoir qu'il existe 2 types de marges :

- **Les marges intérieures**
- **Les marges extérieures**

Regardez bien ce schéma :



Sur ce block, j'ai mis une bordure pour qu'on repère mieux ses bords.

L'espace entre le texte et la bordure est la marge intérieure (en vert).

L'espace entre la bordure et le prochain block est la marge extérieure (en rouge).

En CSS, on peut modifier la taille des marges avec les 2 propriétés suivantes :

- **padding** : indique la taille de la marge intérieure. A exprimer en général en pixels (px).
- **margin** : indique la taille de la marge extérieure. Là encore, on utilise le plus souvent des pixels.

Pour bien voir les marges, prenons 2 paragraphes auxquels je mets simplement une petite bordure :

```
p
{
  width: 350px;
  border: 1px solid black;
  text-align: justify;
}
```

Comme vous pouvez le constater, **il n'y a par défaut pas de marge intérieure (padding)**. En revanche, **il y a une marge extérieure (margin)**. C'est cette marge qui fait que 2 paragraphes ne sont pas collés et qu'on a l'impression de "sauter une ligne". Les marges par défaut ne sont pas les mêmes pour toutes les balises block(et sur tous les navigateurs). **Essayez d'appliquer ce CSS à des balises <div> qui contiennent du texte par exemple, vous verrez que là il n'y a par défaut ni marge intérieure, ni marge extérieure !**

margin et padding s'appliquent aux 4 côtés du block. Si vous voulez indiquer une marge en haut, en bas, à gauche et à droite, il va falloir utiliser des propriétés plus précises .

Voici la liste pour margin :

- **margin-top** : marge extérieure en haut.
- **margin-bottom** : marge extérieure en bas.
- **margin-left** : marge extérieure à gauche.
- **margin-right** : marge extérieure à droite

Et la liste pour padding :

- **padding-top** : marge intérieure en haut.
- **padding-bottom** : marge intérieure en bas.
- **padding-left** : marge intérieure à gauche.
- **padding-right** : marge intérieure à droite.

```
p
{
  width: 350px;
  border: 1px solid black;
  text-align: justify;
  padding: 12px;
  margin-bottom: 50px;
}

h1
{
  margin-left: 30px;
}
```


➤ Centrer des blocks

Il est tout à fait possible de centrer des blocks, c'est même très pratique pour réaliser un design centré quand on ne connaît pas la résolution du visiteur. **Toutefois, et c'est très important, cela ne fonctionne que si vous avez indiqué une largeur précise (width) au block.** Prenons le cas de nos petits paragraphes. On leur a déjà donné une largeur précise ; maintenant si on veut les centrer sur notre écran nous allons mettre la valeur "auto" à la propriété **margin**, comme ceci :

```
p
{
width: 350px; /* On a indiqué une largeur (obligatoire) */
border: 1px solid black;
text-align: justify;
padding: 12px;
margin: auto; /* On peut donc demander à ce que le block soit centré avec "auto" */
margin-bottom: 20px;
}
```

3. Le positionnement en CSS

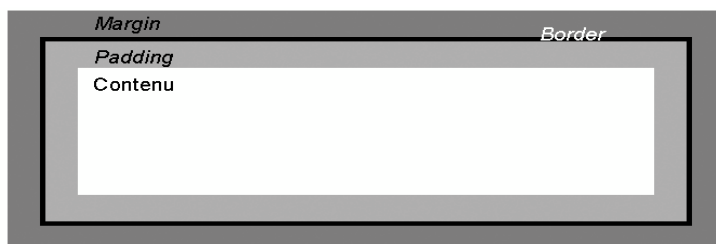
Le placement des objets d'un document web en constitue souvent la charpente. Nous expliquerons par étapes les bases du **positionnement CSS**, pour lesquelles il est nécessaire de bien comprendre le « **modèle de boîte** » **normalisé par le W3C**.

➤ Introduction au positionnement en CSS

L'arrivée de CSS 2 a généralisé le recours aux feuilles de styles. Celles-ci proposent des mises en page bien plus souples que les antiques méthodes à base de tableaux. Contrairement aux cellules de ces derniers, nécessairement adjacentes, les éléments HTML stylés en CSS peuvent facilement être placés n'importe où. Signalons un autre avantage de taille : **la notion de profondeur. CSS travaille sur différents niveaux de profondeur qui se superposent à la manière de couches transparentes. On les appelle souvent calques. Les objets ne sont pas juxtaposés, mais bien souvent superposés.**

➤ Comprendre le modèle de boîte

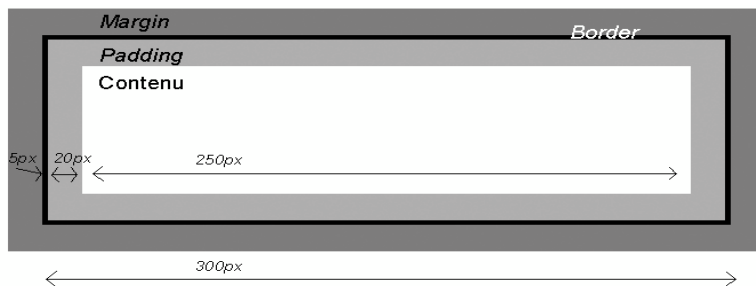
En HTML, matière première de CSS, chaque élément est considéré comme une boîte (voir figure 7-1). En effet, aux dimensions induites par leur contenu s'ajoutent souvent les espaces de marges externes (margin) ou internes (padding) et une bordure (border).



Ces 3 périmètres ne sont pas obligatoires. Non renseignés, ils prendront la valeur par défaut, toujours **nulle pour les éléments en ligne**. En revanche, parmi les éléments de type bloc, **seul <div> n'a pas de marges par défaut**. On peut s'en rendre compte en créant deux boîtes de paragraphes :

➤ Les dimensions des boîtes

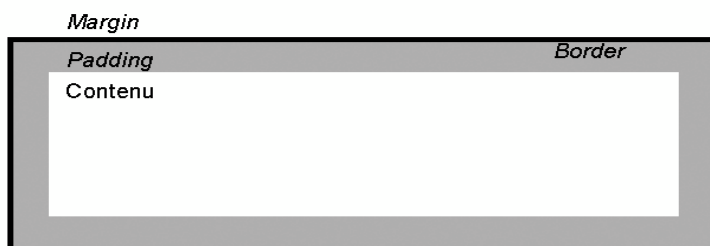
La largeur et la hauteur d'une boîte sont calculées en ajoutant celles du contenu aux dimensions des marges internes et à l'épaisseur de la bordure. Ainsi, la largeur totale d'un élément prendra en compte la largeur du contenu (width), les marges internes gauche et droite, et les épaisseurs des bordures gauche et droite. Il en va de même pour la hauteur (height). Comme le montre la figure 7-2, un élément dont le contenu s'étend sur 250 pixels de large et arborant des marges internes (padding) latérales de 20 pixels et des bordures latérales de 5 pixels occupera à l'écran une largeur totale de : $250 + 20 + 20 + 5 + 5 = 300$ pixels.



➤ Différents modèles de boîtes

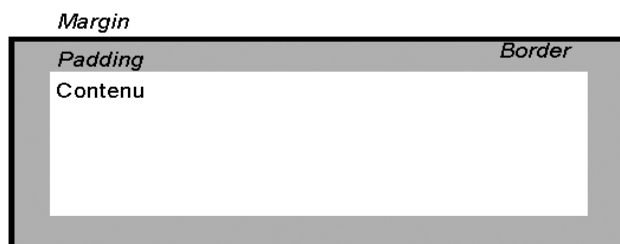
Le navigateur Internet Explorer (dans ses versions 5 et, sous certaines conditions, 6) interprète différemment les dimensions des boîtes. Son éditeur (Microsoft) se fonde sur un modèle non conforme aux recommandations du W3C, et considère pour sa part que le remplissage et les bordures relèvent de la zone de contenu (voir figure 7-3). Dans ces conditions, la largeur apparente d'une boîte sera identique à la largeur spécifiée pour son contenu.

Boîte standard : width 250px



Largeur à l'écran : width + padding + border

Boîte Microsoft : width 250px



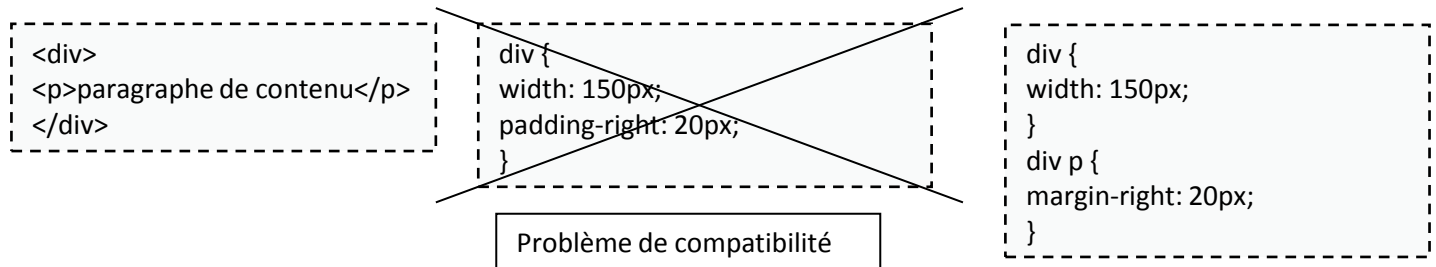
Largeur à l'écran : width

La boîte évoquée plus haut n'occupera alors à l'écran qu'une largeur de 250 pixels. Sa zone de contenu se trouvera donc amputée des épaisseurs de bordures et de remplissages, correspondant ici à une différence de 50 pixels par rapport au modèle standard.

s'expliquent ainsi, les décalages se cumulant d'épaisseur de bordure en épaisseur de padding. Pour éviter nombre de ces ennuis de dimensions, on évitera par conséquent :

- d'attribuer une largeur explicite (width) à un élément doté de marges internes (padding) ou de bordures latérales ;
- d'attribuer une hauteur explicite (height) à un élément doté de marges internes (padding) ou de bordures horizontales. On préférera dans les deux cas recourir aux marges externes (margin).

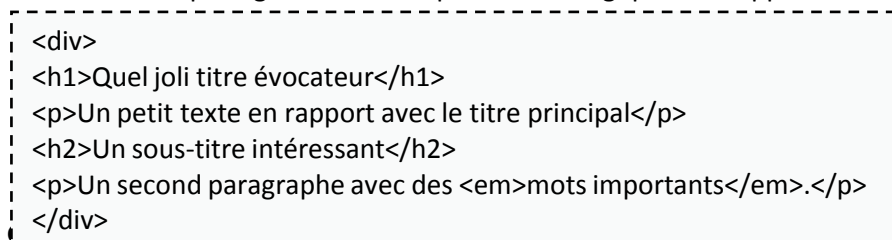
Ainsi, pour un bloc large de 150 pixels qu'on souhaite écarter du reste du document par 20 pixels à droite, il faut recourir à la propriété **padding-right** posera des problèmes de modèle de boîte. Il convient de préférer à une telle marge interne la propriété **margin-right**.



➤ Éléments ancêtres, parents, enfants et frères

Chaque boîte peut en contenir d'autres. Un paragraphe <p> peut ainsi renfermer des boîtes définies par les éléments ou , et se trouver lui-même inclus dans un élément <div>. Toutes ces imbrications de boîtes forment une hiérarchie arborescente entièrement comprise dans la boîte de l'élément racine du document (<body>). Il est essentiel de comprendre cette arborescence pour bien utiliser les positionnements. Tout document HTML est composé de conteneurs (boîtes en renfermant d'autres), sur lesquels la hiérarchie du document induit une généalogie.

- Un élément directement contenu dans un autre est considéré comme son « enfant » (le contenant s'appelle le « parent »). Les éléments de liste sont ainsi enfants d'un élément ou ; un <div> contenant directement un paragraphe <p> en est le parent.
- Les termes familiaux habituels s'en déduisent : un bloc <div> sera l'« ancêtre » de tout élément en ligne (ou) compris dans ses paragraphes enfants. Un parent est un ancêtre de premier niveau. L'ensemble de la lignée ascendante d'un élément, qui aboutit à la racine du document, constitue aussi celui de ses ancêtres.
- Les éléments partageant le même parent sont logiquement appelés « frères ».



- Le bloc <div> est parent des titres <h1>, <h2> et des deux paragraphes <p>. C'est aussi un ancêtre de l'élément .
- Le second bloc <p> est parent de l'élément .
- Les titres <h1>, <h2> et les deux paragraphes <p> sont frères.

Cette hiérarchie structurante permet une mise en page plus fine. On écrira de même les feuilles de styles de manière hiérarchique. Elle présente un autre avantage : un document bien hiérarchisé doit s'afficher de manière tout à fait lisible et fonctionnelle, même en l'absence de styles CSS (et donc de mise en page) – par exemple s'ils sont désactivés.

➤ Comprendre la notion de flux du document

Les différents éléments d'une page, emboîtés ou juxtaposés selon qu'ils sont respectivement parents et enfants ou frères, prennent par défaut place dans le « flux courant » du document, aussi appelé « flux normal ». Il correspond à l'ordre dans lequel les boîtes apparaissent dans le texte, qui est aussi celui de leurs balises dans le code HTML. C'est le cas pour tous les documents HTML, qu'ils soient écrits manuellement ou générés automatiquement. L'ordre du flux courant intervient dans l'affichage par défaut, sans styles, sans mise en page ni positionnement particulier, mais certaines propriétés CSS permettent de « sortir » des éléments du flux courant pour les positionner de façon personnalisée. Dans le flux courant, deux paragraphes (balises de type bloc) s'afficheront par défaut l'un sous l'autre. Adopter un positionnement absolu ou flottant permettra d'extraire l'un d'eux du flux (disons, le second) pour l'afficher ailleurs, par exemple à côté du premier.

➤ Positionner les éléments en CSS

✓ Positionner dans le flux courant

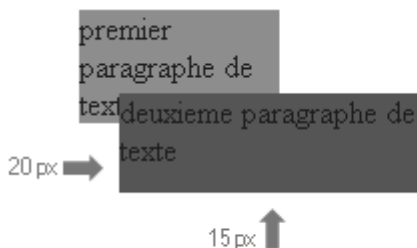
Voici le comportement et le placement spécifiques des éléments dans le flux normal :

- Les éléments de type bloc se succèdent verticalement, chaque nouveau bloc se plaçant sous le bloc frère précédent. Les blocs occupent toute la largeur disponible dans leur conteneur.
- Les éléments en ligne se suivent sur la même ligne, chaque nouvel élément se plaçant directement à la suite du précédent, avec retour à la ligne quand il n'y a plus de place dans le conteneur. Par défaut, chaque élément est donc dépendant de ses frères immédiats, et deux paragraphes `<p>` successifs apparaissent l'un sous l'autre. Cette méthode, appelée positionnement dans le flux courant, convient à la majorité des cas : il suffit simplement de définir les marges de chaque élément pour le placer dans son conteneur.

✓ Le positionnement relatif

C'est une variante du positionnement dans le flux courant, qu'on active par la déclaration **position: relative;**. L'élément concerné est alors dit « positionné », et prend d'abord sa place dans le flux courant. Il peut ensuite s'en décaler à l'aide des propriétés **top**, **right**, **bottom** et **left**.

```
<div>
<p id="premier">premier paragraphe de
texte</p>
<p id="second">deuxième paragraphe de
texte</p>
</div>
```



```
p#premier {
margin-bottom: 0;
margin-top: 0px;
margin-left: 3em;
width: 100px;
background: orange;
}
```

```
p#second {
position: relative;
left: 20px;
bottom: 15px;
margin-top: 0;
margin-left: 3em;

width: 150px;
height: 50px;
background: green;
}
```

Cet exemple illustre l'apport du positionnement relatif, où un élément peut prendre place par-dessus des éléments frères. C'est toutefois une technique à manier avec précaution : on n'y manipule que des décalages, chaque élément restant dépendant de ses frères.

Les techniques classiques de positionnement par marges répondent à la majorité des besoins. Le positionnement relatif n'est vraiment nécessaire que pour des décalages avec superposition. Un élément « positionné » (ici, en relatif) présente aussi des intérêts pour les positionnements absolu et fixe. Positionner un élément lui permet de devenir conteneur d'autres éléments de contenu positionnés.

MISE EN PAGE Impact du positionnement relatif

L'application du positionnement relatif à un élément n'affecte pas les boîtes qui l'entourent ; sa place réservée est celle du positionnement dans le flux normal.

✓ Les positionnements absolu et fixe

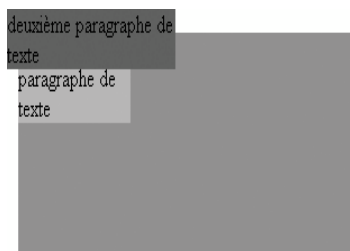
Ils s'appliquent à tout élément doté des déclarations **position: absolute** ou **position: fixed**, qui sort alors du flux pour devenir « positionné ». Les boîtes ainsi retirées du flux normal n'ont plus aucun effet sur le calcul des placements des autres éléments de la même fratrie, et on détermine leur position par les propriétés **top**, **right**, **bottom** et **left**. Ces dernières s'interprètent comme en positionnement relatif et correspondent à des décalages. Toutefois, la position de base est calculée par rapport à un bloc conteneur et non pas en décalage de la position théorique dans le flux normal. **Le boîte conteneur de référence est celle du premier ancêtre positionné (en relatif, fixe ou absolu).**

▪ Le positionnement absolu

Un élément positionné en absolu est donc placé par rapport à son parent si ce dernier est lui-même positionné, et sinon par rapport au premier ancêtre positionné, en remontant au besoin jusqu'à la page entière (élément <body>). Ce principe fondamental distingue le positionnement absolu du placement dans le flux :

- Un élément positionné dans le flux (en non absolu) prend pour conteneur son parent le plus proche. Sa position dépend alors de ses propres marges et du padding de son conteneur.
- Un élément positionné en absolu se reporte au premier ancêtre positionné, qui joue alors le rôle de conteneur.

```
<div>
<p id="premier">premier paragraphe de
texte</p>
<p id="second">deuxième paragraphe de
texte</p>
</div>
```



```
div {
width: 300px;
height: 200px;
background: yellow;
}
p {
margin: 0;
}
p#premier { width:
100px; background:
orange;
}
p#second {
position: absolute;
top: 0;
left: 0;
width: 150px;
height: 50px;
background: green;
}
```

Le second paragraphe prend place en haut à gauche du document, sortant même de la boîte de son parent le bloc <div>. En remplaçant la règle left: 0 par right: 0, on obtient



Il est clair que le second paragraphe ne se place pas en fonction de son parent direct le bloc <div>, mais par rapport aux extrémités du document entier. En effet, un bloc absolu se place selon le premier ancêtre positionné, quitte à remonter jusqu'à la racine du document (l'élément <body>), toujours considérée comme positionnée. Pour placer le second paragraphe par rapport au bloc <div>, il suffit de positionner ce dernier avec une propriété position (en absolu, relatif ou fixe) :



Les propriétés top, right, bottom et left n'ont de sens qu'appliquées à un élément positionné. Elles n'auront aucun effet sur les éléments placés en flux ou flottants.

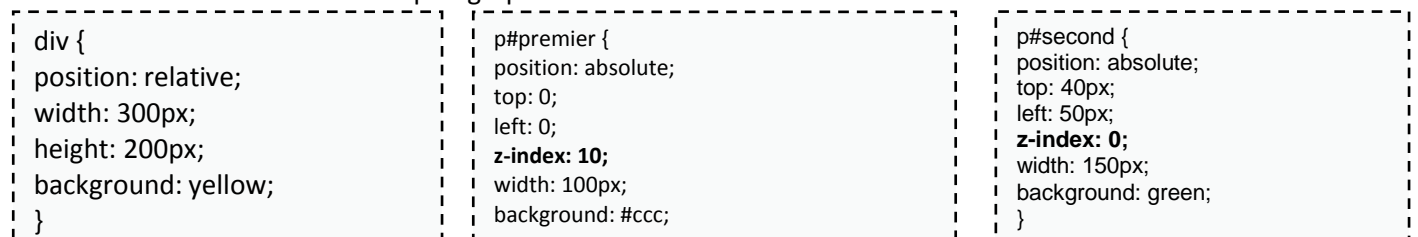
▪ Le positionnement fixe

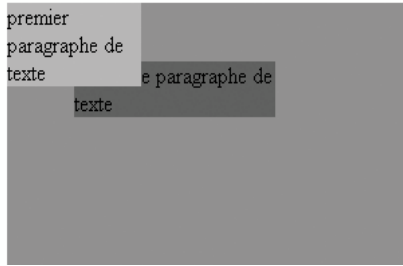
C'est un cas particulier du positionnement absolu, où l'élément reste fixe dans la page, par rapport à la zone de visualisation : il ne se déplace pas lors du défilement de cette dernière. On peut rapprocher cette technique de la propriété **background-attachment pour une image**. Elle permet de simuler le comportement des cadres (frames) sans en avoir les inconvénients en termes d'accessibilité et de lourdeur de structure... mais ce n'est pas une méthode utile pour les mises en pages avec cadres. Malheureusement, elle n'est pas prise en charge par le navigateur Internet Explorer jusqu'à sa version 6. IE7 corrige enfin cette lacune.

✓ La profondeur : z-index

En positionnant des éléments, on peut superposer différents blocs. Ceci ouvre une troisième dimension : celle de la **profondeur**. Par défaut, le dernier élément positionné déclaré dans le code HTML s'affichera par-dessus tous les autres éléments du même conteneur. La propriété **z-index** permet de changer de comportement. Dans un même conteneur, l'élément placé au dessus des autres sera l'élément positionné portant le **plus grand** z-index.

Positionnons absolument les deux paragraphes désormais familiers dans leur conteneur <div>





Les valeurs de z-index n'ont aucune interprétation en absolu. Seule compte leur comparaison, l'élément de valeur la plus élevée prenant place au sommet des zones de superposition.

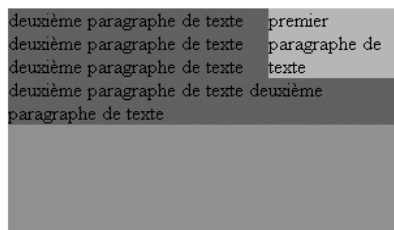
N'abusez pas de ces changements de profondeur : elles sont rarement nécessaires dans un document bien construit et doté d'une structure HTML cohérente.

✓ Le positionnement flottant

On positionne un élément en flottant avec une déclaration **float: left** ou **float: right**. Il est alors retiré du flux normal pour prendre place à gauche (respectivement à droite) du bloc qui le contient : c'est devenu un « flottant ».

L'élément qui le suit s'écoulera alors dans l'espace ainsi laissé libre, en épousant sa forme. Pour mettre cet effet en évidence, il faut allonger le second paragraphe de l'exemple précédent et l'autoriser à occuper toute la largeur disponible :

```
<div>
<p id="premier">premier paragraphe de texte</p>
<p id="second">deuxième paragraphe de texte
deuxième paragraphe de texte
deuxième paragraphe de texte deuxième paragraphe
de texte deuxième
paragraphe de texte </p>
</div>
```



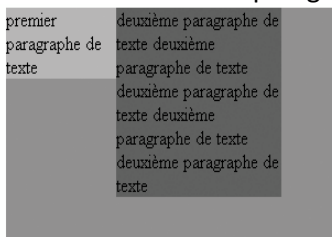
```
div {
width: 300px;
height: 200px;
background: yellow;
}

p#premier { width:
100px; background:
orange; float: right ;
}
p#second {
background: green;
}
```

Le premier paragraphe une fois poussé à droite du conteneur, la suite obéit au flux normal : elle commence au début du conteneur mais s'écoule autour du bloc flottant.

Un des nombreux avantages du positionnement flottant est la possibilité de placer des blocs côte à côte.

Transformer les deux paragraphes de l'exemple en flottants les fait ainsi apparaître au même niveau.



```
p#premier {
float: left;
width: 100px;
background: #ccc;
}
```

```
p#second {
float: left;
width: 150px;
background: green;
}
```

Les deux paragraphes se trouvant désormais hors du flux normal, le bloc <div> est vide de tout contenu. Pour l'instant, il a reçu une hauteur explicite : height: 200px;

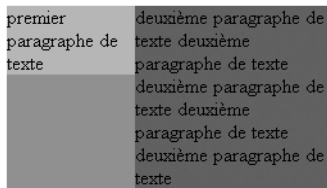
En l'absence d'une telle spécification, quel serait son comportement ? Réduisons cette déclaration à 20 px, ce qui permettra d'observer le comportement du fond jaune. Sur Internet Explorer, le bloc <div> est alors « poussé » par le paragraphe le plus long, mais c'est une mauvaise interprétation du comportement standard. Celui-ci est observable sur d'autres navigateurs, comme Opera ou Firefox. Les deux paragraphes, désormais sortis du flux, n'interfèrent plus sur la hauteur du bloc <div>. Sans le petit résidu de hauteur qu'on lui a accordé, il serait donc invisible.

Ce comportement des éléments flottants est normal mais pose souvent des problèmes. On recourra alors à la propriété clear, qui interdit le voisinage d'un flottant. Ses trois valeurs possibles (left, right, both) portent respectivement sur les côtés gauche, droit, gauche et droit. L'instruction clear: left interdit tout flottant à gauche ; clear: both interdit tout flottant au même niveau. L'élément concerné est alors poussé vers le bas jusqu'à ce qu'il satisfasse les conditions qui lui sont imposées. Testons cette propriété en plaçant une ligne horizontale (<hr/>) sous les flottants. Pour cela, elle devra recevoir la propriété clear: both ;

```
.separation {
clear: both;
}
```

```
<div>
<p id="premier">premier paragraphe de texte</p>
<p id="second">deuxième paragraphe de texte deuxième paragraphe de texte
deuxième paragraphe de texte deuxième paragraphe de texte </p>
<hr class="separation" />
</div>
```

On'oublie pas de supprimer la hauteur explicite du bloc <div> pour que celle-ci s'adapte automatiquement à son contenu. Le conteneur s'est bien étiré jusqu'à inclure la ligne horizontale. On peut ensuite la cacher à l'aide de la déclaration **visibility: hidden**, précisant de masquer l'élément concerné sur les navigateurs graphiques. Grâce à cet élément de séparation, le bloc <div> englobe désormais les deux paragraphes flottants.



Utilisée à bon escient, la propriété float permet de nombreuses fantaisies visuelles, dont la plus fréquente est sans doute de placer une illustration au sein d'un texte de contenu, mise en page fréquente dans les livres ou la presse. C'est aussi à cette propriété qu'on recourt pour créer des lettrines de texte en CSS.

Texte avec lettrine sur 2 lignes texte avec lettrine sur 2 lignes
 texte avec lettrine sur 2 lignes texte avec lettrine sur 2 lignes
 texte avec lettrine sur 2 lignes texte avec lettrine sur 2 lignes
 avec lettrine sur 2 lignes texte avec lettrine sur 2 lignes
 sur 2 lignes texte avec lettrine sur 2 lignes

Texte avec lettrine différente sur 3 lignes texte avec lettrine
 différente sur 3 lignes texte avec lettrine différente sur 3 lignes
 avec lettrine différente sur 3 lignes texte avec lettrine
 différente sur 3 lignes texte avec lettrine différente sur 3 lignes
 avec lettrine différente sur 3 lignes texte avec lettrine différente sur 3 lignes
 avec lettrine différente sur 3 lignes texte avec lettrine différente sur 3 lignes
 avec lettrine différente sur 3 lignes

À nouveau, tout élément positionné en flottant est considéré de type bloc. Les éléments en ligne peuvent ainsi recevoir des dimensions et des bordures.

SYNTHÈSE Fonctionnement du positionnement flottant

- 1 L'élément est placé normalement dans le flux, sous l'éventuel bloc qui le précède et par dessus l'éventuel bloc qui le suit.
- 2 L'élément doté de la propriété float est ensuite « poussé » à gauche ou à droite de son conteneur. Dans ce cas, les éléments qui le suivent dans son conteneur prennent

Mode de positionnement	Conteneurs et blocs	Contenu et éléments internes	Centrage d'éléments	Spécificités
Dans le flux courant	Non	Oui	Oui	Reste dans le flux. Se positionne par rapport aux éléments parent et frères.
Position relative	Non	Oui	Oui	Décalage par rapport au flux avec les propriétés top, right, bottom et left.
Position fixe	Oui	Non	Oui	Sort du flux. Pose des problèmes sur Internet Explorer inférieur à IE7.
Position absolue	Oui	Non	Oui	Sort du flux. Se place par rapport au premier ancêtre positionné. Permet les superpositions.
Placement flottant	Oui	Oui	Non	Sort du flux. Le flottant « centré » n'existe pas. Attention à certains défauts d'affichage.

CHAP 7 : LES TABLEAUX

➤ Un tableau simple

Première balise à connaître : **<table> </table>**. C'est cette balise qui permet d'indiquer le début et la fin d'un tableau. Cette balise est de type block, donc **il faut la mettre en dehors d'un paragraphe**.

Pour commencer en douceur, voici 2 nouvelles balises très importantes :

- **<tr> </tr>** : indique le début et la fin d'une ligne du tableau.
- **<td> </td>** : indique le début et la fin du contenu d'une cellule.

En XHTML, votre tableau se construit ligne par ligne. Dans chaque ligne (<tr>), on indique le contenu des différentes cellules (<td>). Schématiquement, notre tableau de tout à l'heure se construit comme ça :

Nom	Age	Pays
Anne	27 ans	France
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis
Ogasaku Nyagatosoka	18 ans	Japon

si je veux faire un tableau à deux lignes, avec 3 cellules par lignes (donc 3 colonnes), je devrai taper ceci :

```
<table>
  <tr>
    <td>Carmen</td>
    <td>33 ans</td>
    <td>Espagne</td>
  </tr>
  <tr>
    <td>Michelle</td>
    <td>26 ans</td>
    <td>Etats-Unis</td>
  </tr>
</table>
```

Un tableau sans CSS c'est tout vide. Alors justement, rajouter des bordures

```
td /* Toutes les cellules des tableaux... */
{
  border: 1px solid black; /* ... auront une bordure de 1px */
}
```

Ce n'est pas encore aussi parfait que ce qu'on voudrait. En effet, on aimerait qu'il n'y ait qu'une seule bordure entre 2 cellules, or ce n'est pas le cas ici.

Heureusement, il existe une propriété CSS spécifique aux tableaux : **border-collapse**, qui signifie "**coller les bordures entre elles**".

Cette propriété peut prendre 2 valeurs :

- collapse : les bordures seront collées entre elles, c'est l'effet qu'on recherche.
- separate : les bordures seront dissociées (valeur par défaut)

```
table
{
  border-collapse: collapse; /* Les bordures du tableau seront collées (plus joli) */
}
td
{
  border: 1px solid black;
}
```

➤ La ligne d'en-tête

Maintenant que l'on a ce qu'on voulait, on va rajouter la ligne d'en-tête du tableau. Dans notre exemple, les en-têtes sont "Nom", "Age" et "Pays".

La ligne d'en-tête se crée avec un **<tr>** comme on a fait jusqu'ici, mais les cellules à l'intérieur sont cette fois des **<th>** et non pas des **<td>** !

```
<table>
  <tr>
    <th>Nom</th>
    <th>Age</th>
    <th>Pays</th>
  </tr>

  <tr>
    <td>Carmen</td>
    <td>33 ans</td>
    <td>Espagne</td>
  </tr>

  <tr>
    <td>Michelle</td>
    <td>26 ans</td>
    <td>Etats-Unis</td>
  </tr>
</table>
```

Comme le nom des cellules est un peu différent pour l'en-tête, il faut penser à mettre à jour le CSS pour lui dire d'appliquer une bordure sur les cellules normales ET sur l'en-tête :

```
table
{
  border-collapse: collapse;
}
td, th /* Mettre une bordure sur les td ET les th */
{
  border: 1px solid black;
}
```

➤ Titre du tableau

Normalement, tout tableau doit avoir un titre. Le titre permet de renseigner rapidement le visiteur sur le contenu du tableau. Pour le faire, on utilise la balise **<caption></caption>**.

```

<table>
  <caption>Passagers du vol 377</caption>
  <tr>
    <th>Nom</th>
    <th>Age</th>
    <th>Pays</th>
  </tr>
  <tr>
    <td>Carmen</td>
    <td>33 ans</td>
    <td>Espagne</td>
  </tr>
  <tr>
    <td>Michelle</td>
    <td>26 ans</td>
    <td>Etats-Unis</td>
  </tr>
</table>

```

➤ Des tableaux plus élaborés

Nous avons appris à faire des petits tableaux simples. Ces petits tableaux suffisent dans la plupart des cas, mais il arrivera que vous ayez besoin de faire des tableaux plus... complexes.

Nous allons voir 2 techniques particulières :

- Pour les gros tableaux, il est possible de les diviser en 3 parties :
 - En-tête
 - Corps du tableau
 - Pied de tableau
- Pour certains tableaux, il se peut que vous ayez besoin de fusionner des cellules entre elles.

✓ Diviser un gros tableau

Si votre tableau est assez gros, vous aurez tout intérêt à la découper en plusieurs parties. Pour cela, il existe des balises XHTML qui permettent de définir les 3 "zones" du tableau :

- L'en-tête (en haut) : il se définit avec les balises **<thead></thead>**
- Le corps (au centre) : il se définit avec les balises **<tbody></tbody>**
- Le pied du tableau (en bas) : il se définit avec les balises **<tfoot></tfoot>**

Que mettre dans le pied de tableau ? Généralement, si c'est un long tableau, vous recopiez les cellules d'en-tête. Ça permet de voir même en bas du tableau à quoi se rapporte chacune des colonnes.

Schématiquement, un tableau en 3 parties se découpe donc comme cela :

Passagers du vol 377				
En-tête du tableau	Nom	Age	Pays	<thead>
	Carmen	33 ans	Espagne	
	Michelle	26 ans	Etats-Unis	
Corps du tableau	François	43 ans	France	<tbody>
	Martine	34 ans	France	
	Jonathan	13 ans	Australie	
	Xu	19 ans	Chine	
Pied du tableau	Nom	Age	Pays	<tfoot>

La seule chose un peu déroutante, c'est qu'il faut mettre les balises dans l'ordre suivant :

1. <thead>
2. <tfoot>
3. <tbody>

```
<table>
  <caption>Passagers du vol 377</caption>

  <thead> <!-- En-tête du tableau -->
    <tr>
      <th>Nom</th>
      <th>Age</th>
      <th>Pays</th>
    </tr>
  </thead>

  <tfoot> <!-- Pied de tableau -->
    <tr>
      <th>Nom</th>
      <th>Age</th>
      <th>Pays</th>
    </tr>
  </tfoot>

  <tbody> <!-- Corps du tableau -->
    <tr>
      <td>Carmen</td>
      <td>33 ans</td>
      <td>Espagne</td>
    </tr>
    <tr>
      <td>Michelle</td>
      <td>26 ans</td>
      <td>Etats-Unis</td>
    </tr>
```

```
<tr>
  <td>François</td>
  <td>43 ans</td>
  <td>France</td>
</tr>
<tr>
  <td>Martine</td>
  <td>34 ans</td>
  <td>France</td>
</tr>
<tr>
  <td>Jonathan</td>
  <td>13 ans</td>
  <td>Australie</td>
</tr>
<tr>
  <td>Xu</td>
  <td>19 ans</td>
  <td>Chine</td>
</tr>
</tbody>
</table>
```

✓ Fusionner des cellules

Sur certains tableaux complexes, vous aurez besoin de "fusionner" des cellules entre elles. Un exemple de fusion ? Regardez, ce tableau liste des films et indique à qui ils s'adressent :

Titre du film	Pour enfants ?	Pour adolescents ?
Massacre à la tronçonneuse	Non, trop violent	Oui
Les bisounours font du ski	Oui, adapté	Pas assez violent...
Lucky Luke, seul contre tous	Pour toute la famille !	

Pour le dernier film, vous voyez que les cellules ont été fusionnées : elles ne font plus qu'un. C'est exactement l'effet qu'on cherche à obtenir.

Pour effectuer une fusion, il faut rajouter un attribut à la balise <td>. Il faut savoir qu'il existe 2 types de fusion :

- La fusion de colonnes : c'est ce que je viens de faire sur cet exemple. La fusion s'effectue horizontalement. On utilisera l'attribut **colspan**.
- La fusion de lignes : là, deux lignes seront groupées entre elles. La fusion s'effectuera verticalement. On utilisera l'attribut **rowspan**.

Comme vous le savez, vous devez donner une valeur à l'attribut (que ce soit colspan ou rowspan). Cette valeur, c'est le nombre de cellules à fusionner entre elles. Sur notre exemple, on a fusionné deux cellules : celle de la colonne "Pour enfants ?", et celle de "Pour adolescents ?". On devra donc écrire : <td colspan="2"> qui signifie : "Cette cellule est la fusion de 2 cellules". Il est possible de fusionner plus de cellules à la fois (3, 4, 5... tant que vous voulez).

```
<table>
<tr>
  <th>Titre du film</th>
  <th>Pour enfants ?</th>
  <th>Pour adolescents ?</th>
</tr>
<tr>
  <td>Massacre à la tronçonneuse</td>
  <td>Non, trop violent</td>
  <td>Oui</td>
</tr>
<tr>
  <td>Les bisounours font du ski</td>
  <td>Oui, adapté</td>
  <td>Pas assez violent...</td>
</tr>
<tr>
  <td>Lucky Luke, seul contre tous</td>
  <td colspan="2">Pour toute la famille !</td>
</tr>
</table>
```

```
<table>
<tr>
  <th>Titre du film</th>
  <td>Massacre à la tronçonneuse</td>
  <td>Les bisounours font du ski</td>
  <td>Lucky Luke, seul contre tous</td>
</tr>
<tr>
  <th>Pour enfants ?</th>
  <td>Non, trop violent</td>
  <td>Oui, adapté</td>
  <td rowspan="2">Pour toute la famille !</td>
</tr>
<tr>
  <th>Pour adolescents ?</th>
  <td>Oui</td>
  <td>Pas assez violent...</td>
</tr>
</table>
```

Vous pouvez maintenant appliquer du style à votre tableau avec les attributs css que nous avons vu.

Cependant il existe de nouveaux attributs css dédiés aux tableaux :

- **caption-side** : indique où doit se trouver le titre du tableau. Cette propriété peut prendre les valeurs suivantes :
 - **top** : le titre sera placé en haut du tableau (par défaut).
 - **bottom** : le titre sera placé en bas du tableau.
 - **left** : le titre sera placé à gauche du tableau.
 - **right** : le titre sera placé à droite du tableau.
- **border-collapse** : on l'a déjà vue tout à l'heure, cette propriété indique que les bordures des cellules seront collées entre elles. On utilise souvent cette propriété car l'effet qu'elle procure est intéressant. Les valeurs possibles sont :

- **separate** : les bordures seront séparées les unes par rapport aux autres (par défaut).
 - **collapse** : les bordures seront collées entre elles.
-
- **vertical-align** : alignement vertical du contenu d'une cellule. Si une cellule a une hauteur importante, il est possible de placer son contenu en haut, en bas ou au milieu de la cellule :
 - **top** : le contenu sera aligné en haut de la cellule
 - **middle** : le contenu sera aligné au milieu de la cellule
 - **bottom** : le contenu sera aligné en bas de la cellule

CHAP 8 : LES FORMULAIRES

Nous allons parler des formulaires. L'idée est simple : vous avez créé un site, et vous aimeriez par exemple que vos visiteurs puissent vous donner leur avis dessus, en cochant des cases, en entrant leurs commentaires, leurs suggestions....

➤ Créer un formulaire

Pour créer un formulaire, on utilise la balise `<form></form>`. C'est la balise principale du formulaire, elle permet d'en indiquer le début et la fin.

```
<p>Texte avant le formulaire</p>

<form>
  <p>Texte à l'intérieur du formulaire</p>
</form>
```

Notez qu'il faut obligatoirement mettre des balises de type block (comme `<p></p>`) à l'intérieur de votre formulaire si vous avez besoin d'écrire du texte dedans.

Il faut savoir qu'un formulaire est fait pour être envoyé. On va prendre un exemple pour que les choses soient claires.

Supposons que votre visiteur vienne de taper un commentaire dans votre formulaire. Ce message doit être envoyé pour que vous puissiez le recevoir, et afin que vous sachiez ce que le visiteur pense de votre site.

Vous devez indiquer 2 attributs à la balise `<form>` :

- **method** : cet attribut indique par quel moyen les données vont être envoyées. Il existe 2 moyens pour envoyer des données sur le web :
 - **method="get"** : c'est une méthode en général assez peu adaptée, car elle est limitée à 255 caractères. La particularité vient du fait que les informations seront envoyées dans l'adresse de la page (`http://...`), mais ce détail ne nous intéresse pas vraiment pour le moment. La plupart du temps, je vous recommande d'utiliser l'autre méthode : "post".
 - **method="post"** : c'est la méthode la plus utilisée pour les formulaires car on peut rentrer un grand nombre d'informations grâce à elle.
- **action** : c'est l'adresse de la page ou du programme qui va traiter les informations (problème n°2). Cette page se chargera de vous envoyer un mail avec le message si c'est ce que vous voulez, ou bien d'enregistrer le message avec tous les autres dans une base de données.
Cela ne peut pas se faire en XHTML / CSS, on utilisera en général un autre langage (ex : **PHP**).

Pour **method**, vous je vais mettre la valeur "post". Pour **action**, je vais taper le nom d'une page spéciale en PHP ("traitement.php"). C'est cette page qui sera appelée lorsque le visiteur cliquera sur le bouton "Envoyer le formulaire".

```
<p>Texte avant le formulaire</p>

<form method="post" action="traitement.php">
  <p>Texte à l'intérieur du formulaire</p>
</form>

<p>Texte après le formulaire</p>
```


➤ Les zones de saisie

Nous allons voir quelles sont les balises XHTML qui nous permettent de rentrer du texte dans un formulaire. Pour commencer, il faut savoir qu'il y a 2 zones de texte différentes :

- **La zone de texte à une ligne** : comme son nom l'indique, on ne peut écrire qu'une seule ligne à l'intérieur :p. Elle sert pour rentrer des textes courts, comme par exemple : "Entrez votre pseudo"
- **La zone de texte multiligne** : cette zone de texte permet d'écrire une quantité importante de texte sur plusieurs lignes, comme par exemple : "Faites une dissertation sur l'utilité du XHTML dans le développement des pays d'Asie du Sud-Est"

✓ Zone de texte à une ligne

Votre pseudo :

Pour insérer une zone de texte à une ligne, on va utiliser la balise `<input />`. On la retrouvera plusieurs fois par la suite dans ce chapitre. A chaque fois, c'est son attribut type qui va changer. Pour une zone de texte à une ligne, on doit taper :

```
<input type="text" />
```

Mais ce n'est pas tout ! Il manque un attribut qui sera très important : c'est le nom de votre zone de texte. En effet, cela vous permettra plus tard (dans le langage PHP) de reconnaître que tel texte est le pseudo du visiteur, tel texte est son mot de passe etc... Il faut donc donner un nom à cette zone de texte, grâce à l'attribut name. Ici, on va supposer qu'on demande au visiteur de rentrer son pseudo :

```
<input type="text" name="pseudo" />
```

Pensez à entourer votre balise `<input />` par une balise block comme `<p></p>` car sinon votre page web ne sera pas valide (cela vous est expliqué dans l'annexe "Le W3C et les standards du web").

✓ Les labels (étiquettes)

Cette zone de texte est bien jolie, mais si votre visiteur tombe dessus il ne sait pas trop ce qu'il doit mettre dedans. On va justement lui expliquer qu'il doit rentrer son pseudo.

Pour indiquer ce que signifie une zone de texte au visiteur, on utilise la balise `<label>` qui entoure le libellé, comme ceci :

```
<form method="post" action="traitement.php">
  <p>
    <label>Votre pseudo</label> : <input type="text" name="pseudo" />
  </p>
</form>
```

Mais ça ne suffit pas. Il faut lier le label avec la zone de texte. Pour ce faire, il faut donner un nom à la zone de texte, non pas avec l'attribut name mais avec l'attribut id (que l'on peut utiliser sur toutes les balises). Pour lier le label au champ, il faut lui donner un attribut for qui a la même valeur que l'id du champ...En un mot comme en cent, ça donne ça :

```
<form method="post" action="traitement.php">
  <p>
    <label for="pseudo">Votre pseudo</label> : <input type="text" name="pseudo" id="pseudo" />
  </p>
</form>
```

✓ Quelques attributs supplémentaires

Il existe quelques autres attributs sur la balise `<input />` qui vous intéresseront sûrement. Il est ainsi possible, si vous en avez besoin, de donner une valeur par défaut à votre zone de texte. Pour faire cela, il vous suffit d'ajouter l'attribut `value` à `<input />` en indiquant quelle valeur vous voulez mettre au départ. Exemple :

`<input type="text" name="pseudo" value="M@teo21" />` .Autre chose : vous pouvez modifier la largeur de votre zone de texte ainsi que le nombre maximal de caractères que l'on peut mettre dedans. La largeur se définit avec **size**. Le nombre maximal de caractères se définit avec **maxlength**.

```
<form method="post" action="traitement.php">
  <p>
    <label for="pseudo">Votre pseudo :</label>
    <input type="text" name="pseudo" id="pseudo" value="M@teo21" size="30" maxlength="10" />
  </p>
</form>
```

➤ Zone de mot de passe

Vous pouvez facilement faire en sorte que la zone de texte se comporte comme une zone "mot de passe", c'est-à-dire une zone où on ne voit pas à l'écran ce qui est tapé dedans. La seule chose que vous avez besoin de changer, c'est l'attribut `type` de `<input />`. Mettez **type="password"**, et le tour est joué !

Je complète mon formulaire. Il demande maintenant au visiteur son pseudo et son mot de passe :

```
<form method="post" action="traitement.php">
  <p>
    <label for="pseudo">Votre pseudo :</label>
    <input type="text" name="pseudo" id="pseudo" value="M@teo21" />

    <br />
    <label for="pass">Votre mot de passe :</label>
    <input type="password" name="pass" id="pass" />

  </p>
</form>
```

➤ Zone de texte multiligne

Voici une zone de texte multiligne :

Comment pensez-vous que je pourrais améliorer mon site ?

Difficile d'améliorer la perfection non ?

Enfin, moi ce que j'en dis...

A toi de voir !

La zone de texte multiligne est une balise existant par paire (contrairement à `<input />`). C'est la balise

`<textarea></textarea>`

A elle aussi, comme à tout autre élément du formulaire, **il faut lui donner un nom avec name, et utiliser un label qui explique de quoi il s'agit.**

```
<form method="post" action="traitement.php">
  <p>
    <label for="ameliorer">Comment pensez-vous que je pourrais améliorer mon site ?</label><br />
    <textarea name="ameliorer" id="ameliorer"></textarea>
  </p>
</form>
```

On peut modifier la taille du textarea de 2 façons différentes :

- **En CSS** : il suffit d'appliquer les propriétés CSS width et height au textarea.
- Avec des attributs : on peut ajouter **les attributs rows** et **cols** à la balise `<textarea>`. Le premier indique le nombre de lignes du textarea, et le second le nombre de colonnes.

```
<form method="post" action="traitement.php">
  <p>
    <label for="ameliorer">Comment pensez-vous que je pourrais améliorer mon site ?</label><br />
    <textarea name="ameliorer" id="ameliorer" rows="10" cols="50"></textarea>
  </p>
</form>
```

➤ Éléments d'options

En plus des zones de saisies, le XHTML vous offre une ribambelle d'éléments d'options à utiliser dans votre formulaire. On va voir dans cette partie :

- **Les cases à cocher**
- **Les zones d'options**
- **Les listes déroulantes**

✓ Les case à cocher

Ceci, mes amis, ce sont des cases à cocher :

- ☐ Frites
- ☐ Steak haché
- ☐ Epinards
- ☐ Huitres

En effet, la balise à utiliser vous la connaissez déjà : c'est `<input />` . On va seulement changer la valeur de son **attribut type** pour mettre **"checkbox"** : **`<input type="checkbox" name="choix" />`**. Rajoutez un `<label>` bien placé, et le tour est joué !

```
<form method="post" action="traitement.php">
  <p>
    Cochez les aliments que vous aimez manger :<br />
    <input type="checkbox" name="frites" id="frites" /> <label for="frites">Frites</label><br />
    <input type="checkbox" name="steak" id="steak" /> <label for="steak">Steak haché</label><br />
    <input type="checkbox" name="epinards" id="epinards" /> <label for="epinards">Epinards</label><br />
    <input type="checkbox" name="huitres" id="huitres" /> <label for="huitres">Huitres</label>
  </p>
</form>
```

Grâce aux label, vous n'êtes pas obligés de cliquer directement sur la case, vous pouvez aussi cliquer sur le texte juste à côté (enfin, ça ne marche pas sur IE)

N'oubliez pas aussi de donner un nom à chaque case à cocher, cela vous permettra d'identifier plus tard quelles cases le visiteur a coché.

Ah si, j'allais oublier. Vous pouvez faire en sorte qu'une case soit déjà cochée par défaut. Pour faire cela, il faut rajouter l'attribut `checked="checked"`

```
<form method="post" action="traitement.php">
  <p>
    Cochez les aliments que vous aimez manger :<br />
    <input type="checkbox" name="frites" id="frites" /> <label for="frites">Frites</label><br />
    <input type="checkbox" name="steak" id="steak" /> <label for="steak">Steak haché</label><br />
    <input type="checkbox" name="epinards" id="epinards" checked="checked" /> <label
for="epinards">Epinards</label><br />
    <input type="checkbox" name="huitres" id="huitres" /> <label for="huitres">Huitres</label>
  </p>
</form>
```

✓ Les zones d'options (boutons radio)

Les zones d'options vous permettent de faire un choix (et un seul) parmi une liste de possibilités :

- ☐ Moins de 15 ans
- ☐ 15-25 ans
- ☐ 25-40 ans

Ca ressemble aux cases à cocher, avec juste une petite difficulté supplémentaire, vous allez voir.

La balise à utiliser est toujours un `<input />`, avec cette fois la valeur "**radio**" pour l'attribut type.

La grosse différence avec les cases à cocher, c'est que les zones d'options fonctionnent par "groupe". **Tout un groupe d'options a le même nom, mais un attribut value différent à chaque fois.**

```
<form method="post" action="traitement.php">
  <p>
    Veuillez indiquer la tranche d'âge dans laquelle vous vous situez :<br />
    <input type="radio" name="age" value="moins15" id="moins15" /> <label for="moins15">Moins de 15
ans</label><br />
    <input type="radio" name="age" value="medium15-25" id="medium15-25" /> <label for="medium15-
25">15-25 ans</label><br />
    <input type="radio" name="age" value="medium25-40" id="medium25-40" /> <label for="medium25-
40">25-40 ans</label><br />
    <input type="radio" name="age" value="plus40" id="plus40" /> <label for="plus40">Encore plus vieux que
ça ?!</label>
  </p>
</form>
```

Pourquoi avoir mis le même nom pour chaque option ? Tout simplement pour que le navigateur sache dans quel "groupe" le bouton fait partie. Essayez d'enlever les attributs name, vous verrez que chaque élément d'option deviendra sélectionnable. Or, ce n'est pas ce que l'on veut, c'est pour ça qu'on les "lie" entre eux en leur donnant un nom identique. Vous noterez que cette fois on a choisi un id différent de name. En effet, les name étant identiques, on n'aurait pas pu les différencier (et vous savez bien qu'un id doit être unique !). Voilà donc pourquoi on a choisi de mettre à l'id la même valeur que value. Si vous avez 2 zones d'options différentes, il faut donner un nom unique à chaque groupe comme ceci :

```
<form method="post" action="traitement.php">
  <p>
    Veuillez indiquer la tranche d'âge dans laquelle vous vous situez :<br />
    <input type="radio" name="age" value="moins15" id="moins15" /> <label for="moins15">Moins de 15 ans</label><br />
    <input type="radio" name="age" value="medium15-25" id="medium15-25" /> <label for="medium15-25">15-25
ans</label><br />
    <input type="radio" name="age" value="medium25-40" id="medium25-40" /> <label for="medium25-40">25-40
ans</label><br />
    <input type="radio" name="age" value="plus40" id="plus40" /> <label for="plus40">Encore plus vieux que ça ?!</label>
  </p>
  <p>
    Sur quel continent habitez-vous ?<br />
    <input type="radio" name="continent" value="europe" id="europe" /> <label
for="europe">Europe</label><br />
    <input type="radio" name="continent" value="afrique" id="afrique" /> <label for="afrique">Afrique</label><br />
    <input type="radio" name="continent" value="asie" id="asie" /> <label for="asie">Asie</label><br />
    <input type="radio" name="continent" value="amerique" id="amerique" /> <label for="amerique">Amérique</label><br />
    <input type="radio" name="continent" value="australie" id="australie" /> <label for="australie">Australie</label>
  </p>
</form>
```

Si, au lieu de mettre `name="continent"` on avait continué à mettre des `name="age"`, le visiteur n'aurait pas pu sélectionner son âge et son continent. Essayez de changer les noms, vous verrez bien ce qu'il se passe. Dernière petite précision : si vous voulez qu'une des options soit cochée par défaut, vous rajoutez un `checked="checked"` comme pour les cases à cocher.

✓ Les listes déroulantes

Les listes déroulantes sont un autre moyen élégant de faire un choix entre plusieurs possibilités :

Dans quel pays habitez-vous ?



Cette fois, ça fonctionne un peu différemment. On va utiliser la balise `<select></select>` qui indique le début et la fin de la liste déroulante.

On ajoute l'attribut `name` à la balise pour donner un nom à la liste. Par exemple "pays" : `<select name="pays">`

Et maintenant, à l'intérieur du `<select></select>`, on va mettre plusieurs balises `<option></option>` (une par choix possible).

On rajoute un attribut `value` pour pouvoir identifier ce que le visiteur a choisi.

```
<form method="post" action="traitement.php">
  <p>
    <label for="pays">Dans quel pays habitez-vous ?</label><br />
    <select name="pays" id="pays">
      <option value="france">France</option>
      <option value="espagne">Espagne</option>
      <option value="italie">Italie</option>
      <option value="royaume-uni">Royaume-Uni</option>
      <option value="canada">Canada</option>
      <option value="etats-unis">Etats-Unis</option>
      <option value="chine">Chine</option>
      <option value="japon">Japon</option>
    </select>
  </p>
</form>
```

Le principe est un peu différent de ce qu'on a vu jusqu'ici, mais ça se comprend néanmoins très bien. Autre nouveauté, on ne peut plus utiliser le `checked="checked"` ici, on doit utiliser à la place le **`selected="selected"`**. Il nous permet comme le `checked` de sélectionner une valeur par défaut :

```

<form method="post" action="traitement.php">
  <p>
    <label for="pays">Dans quel pays habitez-vous ?</label><br />
    <select name="pays" id="pays">
      <option value="france">France</option>
      <option value="espagne">Espagne</option>
      <option value="italie">Italie</option>
      <option value="royaume-uni">Royaume-Uni</option>
      <option value="canada" selected="selected">Canada</option>
      <option value="etats-unis">Etats-Unis</option>
      <option value="chine">Chine</option>
      <option value="japon">Japon</option>
    </select>
  </p>
</form>

```

Mais les listes d'options savent faire encore mieux ! On peut créer des groupes d'options à l'intérieur de la liste, grâce à la balise **<optgroup></optgroup>**. Vous devez lui ajouter l'attribut label qui permet de donner un nom au groupe (à ne pas confondre avec la balise <label> !).

Dans notre exemple, pourquoi ne pas séparer les pays en fonction de leur continent ?

```

<form method="post" action="traitement.php">
  <p>
    <label for="pays">Dans quel pays habitez-vous ?</label><br />
    <select name="pays" id="pays">
      <optgroup label="Europe">
        <option value="france">France</option>
        <option value="espagne">Espagne</option>
        <option value="italie">Italie</option>
        <option value="royaume-uni">Royaume-Uni</option>
      </optgroup>
      <optgroup label="Amérique">
        <option value="canada">Canada</option>
        <option value="etats-unis">Etats-Unis</option>
      </optgroup>
      <optgroup label="Asie">
        <option value="chine">Chine</option>
        <option value="japon">Japon</option>
      </optgroup>
    </select>
  </p>
</form>

```

➤ Un formulaire accessible et design ?

Maintenant, on va essayer d'aller encore plus loin. Notre objectif sera double : faire en sorte que notre formulaire soit accessible et design

On va faire ça en 4 étapes :

1. Définir un ordre de tabulation (accessibilité)

2. Définir des touches de raccourci (accessibilité)
3. Organiser le formulaire en plusieurs zones (accessibilité et design)
4. Rajouter du CSS (design)

✓ Définir un ordre de tabulation

C'est le premier des points que nous allons voir censé nous faciliter la vie. Comme vous le savez peut-être, on peut se déplacer dans un formulaire uniquement grâce à la touche "Tab" (tabulation) située à gauche de votre clavier. A chaque fois qu'on appuie sur Tab, on va au champ suivant. A chaque fois qu'on fait Maj + Tab, on retourne au champ précédent. Le but est de dire en XHTML dans quel ordre on doit se déplacer dans le formulaire. Par exemple, après le champ "nom" si je tape Tab je dois tomber sur le champ "prénom", puis sur "e-mail" etc...

On va utiliser l'attribut **tabindex** qui peut se rajouter sur toutes les balises du formulaire qu'on a apprises.

On doit lui mettre un nombre pour valeur. Chaque champ du formulaire doit avoir un nombre différent.

Les nombres indiquent dans quel ordre on se déplace dans le formulaire : d'abord le n°1, puis le n°2, le n°3 etc...

Vous n'êtes pas obligés de mettre des nombres qui se suivent. Il est même conseillé de laisser des "espaces" entre les nombres au cas où vous auriez besoin de rajouter plus tard des champs.

Ainsi, il est tout à fait possible de compter 10 par 10 : n°10, n°20, n°30 etc... Ca ne coûte pas plus cher de compter de 10 en 10, et si plus tard on a besoin de créer un champ n°25, on n'aura aucun problème .

Par défaut, si aucun tabindex n'est mis, le navigateur dira que le premier champ est celui tout en haut, et que le dernier celui tout en bas de la page.

Cependant, je vous conseille de toujours mettre vous-même les tabindex, car si votre formulaire se complexifie par la suite cela sera très utile.

✓ Définir des touches de raccourci

Une touche de raccourci ("access key" en anglais) est une touche qui permet d'accéder directement à un champ de votre formulaire sans avoir à cliquer dessus avec la souris et sans avoir à appuyer plusieurs fois sur "Tab" comme un forcené avant de tomber sur le champ qui vous intéresse.

Ce qui est très pratique, c'est que le XHTML vous permet de choisir quelles touches du clavier serviront de raccourcis.

Ce qui l'est moins, c'est que les raccourcis s'utilisent de manière différente en fonction du navigateur :

- Firefox et Internet Explorer (Windows) : il faut faire la combinaison de touches **Alt + Raccourci**. **Si ça ne marche pas, essayez Alt + Maj + Raccourci.**
- Safari et IE-Mac (Macintosh) : il faut taper **Ctrl + Raccourci**.

Pour définir une touche de raccourci, vous utiliserez l'attribut **accesskey** qui, comme tabindex, peut se mettre sur tous les types de champs de formulaire qu'on a vus. Vous devez lui mettre comme valeur la touche du clavier qui doit servir de raccourci pour le champ.

Le gros problème des touches de raccourci est que certains caractères sont déjà utilisés par le navigateur. Si vous utilisez les mêmes, il y aura un conflit et vos visiteurs ne pourront plus utiliser les raccourcis auxquels ils sont habitués. L'idéal est d'utiliser des chiffres en raccourci, ils sont en général très peu utilisés par les navigateurs. N'oubliez pas d'indiquer quelque part sur la page quels sont les raccourcis utilisables, parce que vos visiteurs ne pourront pas les deviner.

➤ Organiser le formulaire en plusieurs zones

La technique que nous allons voir a 2 avantages :

- Elle permet de rendre le formulaire plus clair, donc plus accessible.
- Elle permet d'améliorer le design de votre formulaire.

Si vous avez un formulaire assez gros (et en général ça sera le cas), il est facile que le visiteur se perde dans la masse d'informations qu'il a à entrer. Il est possible en XHTML de grouper plusieurs champs ayant un thème entre eux. On utilisera la balise **<fieldset></fieldset>** pour délimiter un groupe de champs. A l'intérieur de cette balise, vous mettez vos champs (vos `<input />` entre autres...) ainsi qu'une autre balise : **<legend></legend>**. Celle-ci permet de donner le nom du groupe. Ça ressemble étrangement aux `<optgroup>` qu'on vient de voir ça... Mais pourquoi tout à l'heure on a utilisé un attribut (label) pour donner un titre au groupe, et là on utilise une balise spéciale `<legend>` ?

```
<form method="post" action="traitement.php">

  <fieldset>
    <legend>Vos coordonnées</legend> <!-- Titre du fieldset -->

    <label for="nom">Quel est votre nom ?</label><br />
    <input type="text" name="nom" id="nom" tabindex="10" /><br />

    <label for="prenom">Quel est votre prénom ?</label><br />
    <input type="text" name="prenom" id="prenom" tabindex="20" /><br />

    <label for="email">Quel est votre e-mail ?</label><br />
    <input type="text" name="email" id="email" tabindex="30" /><br />
  </fieldset>

  <fieldset>
    <legend>Votre souhait</legend> <!-- Titre du fieldset -->

    <p>
      Faites un souhait que vous voudriez voir exaucé :<br />
      <input type="radio" name="souhait" value="riche" id="riche" tabindex="40" /> <label for="riche">Etre
riche</label><br />
      <input type="radio" name="souhait" value="celebre" id="celebre" tabindex="50" /> <label
for="celebre">Etre célèbre</label><br />
      <input type="radio" name="souhait" value="intelligent" id="intelligent" tabindex="60" /> <label
for="intelligent">Etre <strong>encore</strong> plus intelligent</label><br />
      <input type="radio" name="souhait" value="autre" id="autre" tabindex="70" /> <label
for="autre">Autre...</label><br />
    </p>

    <p>
      <label for="precisions">Si "Autre", veuillez préciser :</label><br />
      <textarea name="precisions" id="precisions" cols="40" rows="4" tabindex="80"></textarea>
    </p>
  </fieldset>
</form>
```

A l'intérieur des `<fieldset></fieldset>`, l'utilisation de balises de paragraphe `<p></p>` n'est plus obligatoire comme c'était le cas tout à l'heure.

➤ Y'a plus qu'à appuyer sur le bouton !

Nous avons vu la plupart des éléments que l'on peut intégrer dans un formulaire, mais il manque le plus important : le bouton de validation ! Heureusement, c'est très facile à créer, d'autant plus que vous connaissez déjà la balise `<input />`. Et il ya 3 type de boutons

- **Le bouton d'envoi** : il déclenche l'envoi du formulaire. Le visiteur se retrouve automatiquement .Un bouton d'envoi se crée avec l'attribut **type="submit"**. Vous pouvez lui ajouter **un attribut value pour changer le texte à l'intérieur du bouton**.
`<input type="submit" value="Envoyer" />`
- **Le bouton de remise à zéro** : il remet à zéro automatiquement toutes les valeurs du formulaire. On doit utiliser cette fois l'attribut **type="reset"**
`<input type="reset" />`
- **Le bouton qui-sert-à-rien** : c'est un bouton "générique" qui n'effectue aucune action particulière. Le formulaire n'est pas envoyé, il n'est pas remis à zéro, non rien ne se passe.
Quel intérêt ? Ca vous servira principalement à lancer des scripts en **Javascript** (un autre langage qu'on peut utiliser sur une page web). **Cette fois, je vous recommande de mettre l'attribut value pour que l'on sache à quoi sert le bouton :**
`<input type="button" value="Je sers à rien" />`

```
<form method="post" action="cible_formulaire.php">

<fieldset>
  <legend>Vos coordonnées</legend>

  <label for="nom">Quel est votre nom ?</label><br />
  <input type="text" name="nom" id="nom" tabindex="10" /><br />

  <label for="prenom">Quel est votre prénom ?</label><br />
  <input type="text" name="prenom" id="prenom" tabindex="20" /><br />

  <label for="email">Quel est votre e-mail ?</label><br />
  <input type="text" name="email" id="email" tabindex="30" /><br />
</fieldset>

<fieldset>
  <legend>Votre souhait</legend>

  <p>
    Faites un souhait que vous voudriez voir exaucé :<br />
    <input type="radio" name="souhait" value="riche" id="riche" tabindex="40" /> <label for="riche">Etre
    riche</label><br />
    <input type="radio" name="souhait" value="celebre" id="celebre" tabindex="50" /> <label for="celebre">Etre
    célèbre</label><br />
    <input type="radio" name="souhait" value="intelligent" id="intelligent" tabindex="60" /> <label
    for="intelligent">Etre <strong>encore</strong> plus intelligent</label><br />
    <input type="radio" name="souhait" value="autre" id="autre" tabindex="70" /> <label
    for="autre">Autre...</label><br />
  </p>
</fieldset>
```

```
<textarea name="precisions" id="precisions" cols="40" rows="4" tabindex="80"></textarea>
  </p>
</fieldset>

<p>
  <input type="submit" /> <input type="reset" />
</p>

</form>
```

Dans le cas présent, le formulaire ne fait strictement rien. Lorsque vous cliquez sur "Envoyer", le formulaire vous amène donc à une page "cible_formulaire.php", qui est une page fictive en PHP que vous ne savez pas faire.