

## Sommaire

<b>Introduction aux feuilles de style</b>	<b>2</b>
Constituants des pages web . . . . .	2
Définition . . . . .	2
Objectifs . . . . .	2
Avantages . . . . .	2
Principe . . . . .	3
<b>Syntaxe CSS</b>	<b>4</b>
Jeu de règles . . . . .	4
Propriétés et valeurs . . . . .	4
Les sélecteurs . . . . .	4
Exemples sur les sélecteurs . . . . .	8
<b>Principes techniques</b>	<b>10</b>
Cascade de CSS . . . . .	10
Ancêtres, Parents, Enfants et Frères . . . . .	10
Notion d'héritage . . . . .	10
Unités de mesures . . . . .	11
Flux normal . . . . .	11
Les dimensions des boîtes . . . . .	14
Position . . . . .	16
Boîte flottante . . . . .	18
<b>Liens</b>	<b>20</b>

Les objectifs de ce support de cours sont de fournir les notions de base pour utiliser efficacement les feuilles de style dans la conception de site web en respectant les bonnes pratiques.



Il existe de très nombreux sites dédiés au CSS ! Il faut au moins l'accès à la documentation du w3c : [www.yoyodesign.org/doc/w3c/css2/](http://www.yoyodesign.org/doc/w3c/css2/).

Ce document a été rédigé à partir de la documentation CSS2 du W3C et d'articles publiés sur les sites : [www.alsacreations.com](http://www.alsacreations.com) et [openweb.eu.org](http://openweb.eu.org).

# Introduction aux feuilles de style

## Constituants des pages web

- La structure et le contenu (statique) : en HTML ou en XHTML
- **La présentation : avec les feuilles de style CSS**
- Le comportement géré coté client par le navigateur : en Javascript
- La navigation et l'échange de données : par l'intermédiaire du protocole HTTP et l'utilisation de Web Service ou d'AJAX
- La génération de contenu (dynamique) coté serveur : avec des langages de développement de type PHP, Java, ...
- Le graphisme par découpage et intégration des images : GIF, JPG, PNG, ...
- L'animation : en Flash ou en SVG, et depuis peu, avec certains attributs du CSS3
- L'incorporation de multimédias : AVI, MPG, MP3, ...

## Définition

Les **feuilles de style en cascade CSS** (*Cascading Style Sheets*) est un **langage informatique qui sert à décrire la présentation des documents HTML, XHTML et XML**. Les standards définissant CSS sont publiés par le W3C (*World Wide Web Consortium*).



Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.

Source : [wikipedia.fr](http://wikipedia.fr)

## Objectifs

L'un des objectifs majeurs des feuilles de style CSS est de **séparer la structure d'un document de ses styles de présentation**.

## Avantages


- La structure du document et la présentation peuvent être gérées dans des fichiers séparés.
- La conception d'un document se fait dans un premier temps sans se soucier de la présentation, ce qui permet d'être plus efficace.
- Dans le cas d'un site web, la présentation est uniformisée : les documents (pages HTML) font référence aux mêmes feuilles de styles. Cette caractéristique permet de plus une remise en forme rapide de l'aspect visuel.
- Un même document peut donner le choix entre plusieurs feuilles de style, par exemple une pour l'impression et une pour la lecture à l'écran.
- Le code HTML est considérablement réduit en taille et en complexité, puisqu'il ne contient plus de balises ni d'attributs de présentation.

## Principe

Il est par exemple possible de ne décrire que la structure d'un document en HTML :

```
<html>
  <head>
    <title>Un titre de document</title>
    <link rel="stylesheet" type="text/css" href="feuille.css" media="screen" />
  </head>
  <body>
    <h1>Un titre de niveau 1 (en bleu)</h1>
    <p>Un paragraphe (en rouge)</p>
  </body>
</html>
```

*La structure d'un document en HTML*

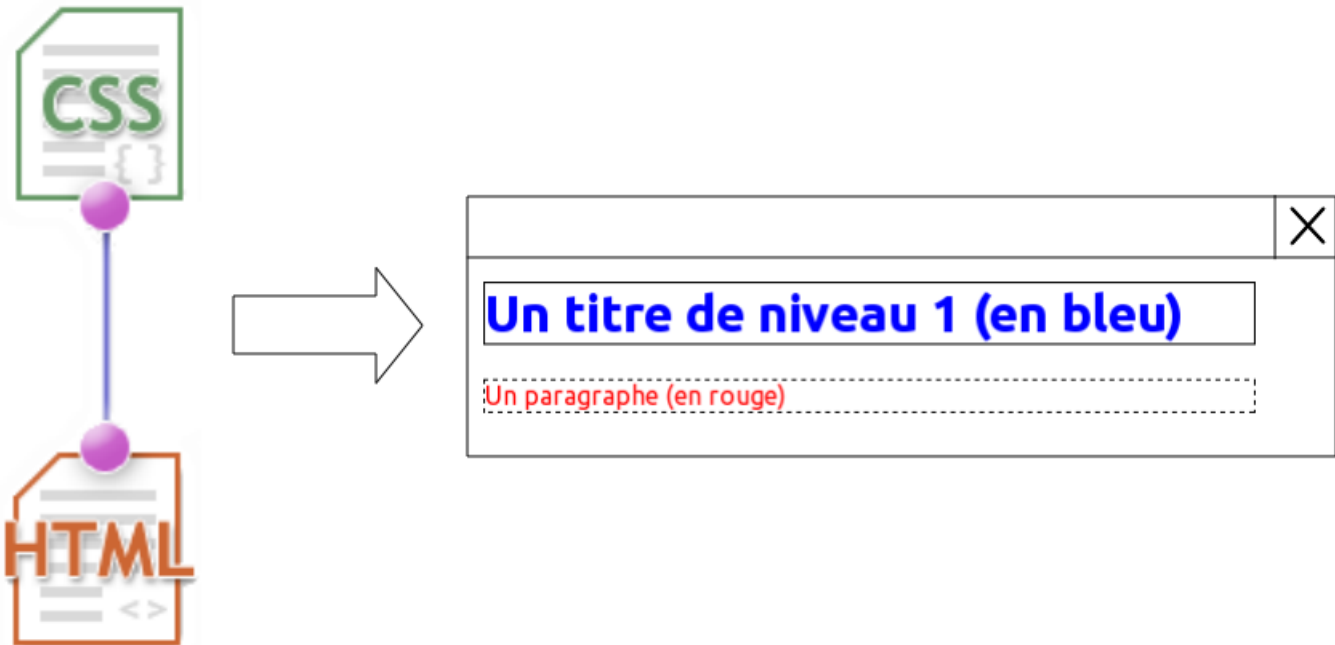
 La déclaration d'une feuille de style CSS se fait dans l'en-tête (partie <head>) du document HTML avec la balise <link>. Il est possible de définir une feuille de style selon le type de média de sortie, voici quelques valeurs possibles de l'attribut media : all (tous), screen (écran), print (imprimante), handheld (PDA), ...


Et de décrire toute la présentation dans une feuille de style CSS séparée :

```
h1 { color: blue; border: solid 1px black; }
p { color: red; border: dashed 1px black; }
```

*La présentation dans une feuille de style CSS*

On obtient l'affichage suivant dans un navigateur :




 La règle '@import' permet l'importation de règles de style à partir d'une autre feuille de style. Les règles @import doivent précéder toutes autres règles dans la feuille de style. Le mot-clé '@import' doit être suivi de l'adresse URI de la feuille de style à intégrer.

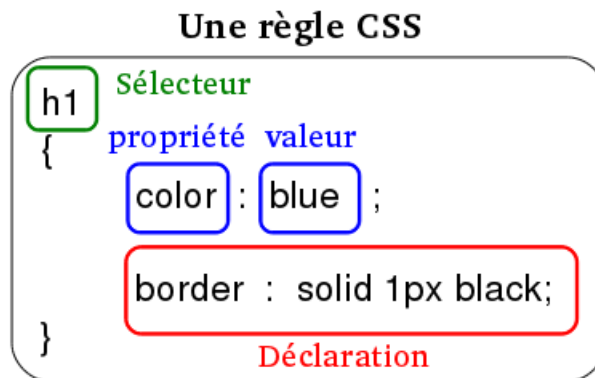
# Syntaxe CSS

## Jeu de règles

Un jeu de règles (qualifié aussi de "règle") se compose d'un **sélecteur** suivi par un **bloc de déclaration** délimité par des accolades (`{}`).

Une déclaration est constituée d'une **propriété**, suivie du caractère deux-points (`:`) puis d'une **valeur**.

 Les multiples déclarations qui se rapportent à un même sélecteur sont séparées par des points-virgules (`;`).




## Propriétés et valeurs

Les caractéristiques de style sont exprimées sous forme de couples **propriété : valeur**.

Les propriétés sont libellées à l'aide de mots-outils anglais tels que « width » (largeur), « font-size » (taille de la police de caractères), ...

*Index des propriétés CSS2* : [www.yoyodesign.org/doc/w3c/css2/propidx.html](http://www.yoyodesign.org/doc/w3c/css2/propidx.html)

 Les commentaires commencent par les caractères `/*` et se terminent par `*/`.

## Les sélecteurs


Un sélecteur est un **motif de reconnaissance permettant d'appliquer un style aux éléments de l'arbre du document HTML**.

– À tous les éléments de la page : le sélecteur universel `*` agit sur tous les éléments HTML.

```
* { color: #ff0000; } /* pour définir une couleur rouge par défaut */
```

– À toutes les instances d'un élément : les sélecteurs de type agissent sur un type d'élément HTML.

```
/* pour définir une couleur bleue pour tous les textes des paragraphes */
p { color: #0000ff; }
```

 Sauf si une autre couleur leur est attribuée par ailleurs de façon plus spécifique.

- À différents éléments simultanément : le regroupement de sélecteurs de type

```
/* pour définir une couleur marron pour tous les titres de niveau 3, 4, 5 et 6 */  
h3,h4,h5,h6 { color: brown; }
```

- À certaines instances d'un élément : les sélecteurs de classe

```
/* pour définir une couleur verte pour les textes de certains paragraphes (ceux de la classe  
.vert) */  
p.vert { color: #008000; }
```



Le contenu de toutes les balises `<p class="vert">` sera vert. Et le contenu des balises `<p>` restera bleu.

- À une instance unique d'un élément : les sélecteurs d'id

```
/* pour définir une couleur grise pour le texte d'un paragraphe précis */  
p#gris { color: #7F7F7F; }  
/* ou */  
#gris p { color: #7F7F7F; }
```



Le contenu de la seule balise `<p id="gris">` sera gris. Le contenu de toutes les balises `<p class="vert">` sera vert. Et le contenu des autres balises `<p>` restera bleu.

- À un ensemble d'éléments successifs : aux éléments bornés par l'élément `div`

```
/* pour définir une couleur verte pour une classe */  
.vert { color: #008000; }
```

```
/*  
qui s'appliquera aussi bien à :
```

```
<div class="vert">  
  <h1>...</h1>  
  <p>...</p>  
  <p>...</p>  
</div>
```

```
ou définis par :
```

```
<h1 class="vert">...</h1>  
<p class="vert">...</p>  
<p class="vert">...</p>  
*/
```

- À une partie de contenu de paragraphe : l'élément `span`

```
/* pour définir une couleur verte pour une classe */
span.vert { color: #008000; }

/*
qui s'appliquera à :

<p>
  <span class="vert">
    ceci sera en vert
  </span>
  ... ceci sera donc en bleu ...
</p>
*/
```

- À un élément directement ou indirectement contenu dans un autre élément : les sélecteurs descendants

```
/* pour définir une couleur rose d'un élément em contenu dans un élément h2 */
h2 em { color: pink; }

/*
qui s'appliquera aussi bien à :

<h2>
  <em>en rose</em>
  ...
</h2>

qu'à :

<h2>
  <code><em>en rose</em></code>
  ...
</h2>
*/
```

- À un élément directement contenu dans un autre élément : les sélecteurs d'enfant

```
/* pour définir une couleur jaune d'un élément em descendant de l'élément p */
p>em { color: yellow; }

/*
qui s'appliquera seulement à :

<p><em>en jaune</em>
  ...
</p>

mais pas à :

<p><q><em>...</em></q>
  ...
</p>
*/
```

– À un élément suivant un autre élément : les sélecteurs d'enfants adjacents

```
/* pour définir une couleur kaki d'un élément p suivant directement un élément img */
img + p { color: khaki; }

/*
qui s'appliquera à :

<img>...</img>
<p>en kaki</p>

mais pas à :

<h3>...</h3>
<p>...</p>
*/
```

Cette table résume la syntaxe du sélecteur de CSS2 :

Motif	Signification	Décrit au chapitre...
*	Correspond à tout élément.	<a href="#">Sélecteur universel</a>
E	Correspond à tout élément E (c.à.d., un élément de type E).	<a href="#">Sélecteurs de type</a>
E F	Correspond à tout élément F qui est un descendant de l'élément E.	<a href="#">Sélecteurs descendants</a>
E > F	Correspond à tout élément F aussi un enfant de l'élément E.	<a href="#">Sélecteurs d'enfant</a>
E:first-child	Correspond à un élément E aussi le premier enfant de son élément parent.	<a href="#">La pseudo-classe :first-child</a>
E:link E:visited	Correspond à un élément E qui est une ancre dans la source dont le lien n'a pas été visité (:link) ou bien l'a déjà été (:visited).	<a href="#">Les pseudo-classes de lien</a>
E:active E:hover E:focus	Correspond à l'élément E au cours de certaines actions de l'utilisateur.	<a href="#">Les pseudo-classes dynamiques</a>
E:lang(c)	Correspond à l'élément de type E qui emploie une langue c (la détermination de cette langue est spécifique au langage du document).	<a href="#">La pseudo-classe :lang()</a>
E + F	Correspond à tout élément F immédiatement précédé par un élément E.	<a href="#">Les sélecteurs adjacents</a>
E[foo]	Correspond à tout élément E avec l'attribut "foo" (quelles qu'en soient les valeurs).	<a href="#">Sélecteurs d'attribut</a>
E[foo="warning"]	Correspond à tout élément E dont l'attribut "foo" a exactement la valeur "warning".	<a href="#">Sélecteurs d'attribut</a>
E[foo~="warning"]	Correspond à tout élément E dont l'attribut "foo" a pour valeur une liste de valeurs séparées par des caractères blancs et dont une de celles-ci est "warning".	<a href="#">Sélecteurs d'attribut</a>
E[lang="en"]	Correspond à tout élément E dont l'attribut "lang" a pour valeur une liste de valeurs séparées par des tirets, cette liste commençant (à gauche) par "en".	<a href="#">Sélecteurs d'attribut</a>
DIV.warning	<i>Seulement en HTML.</i> Identique à DIV[class~="warning"].	<a href="#">Sélecteurs de classe</a>
E#myid	Correspond à tout élément E dont l'ID est "myid".	<a href="#">Sélecteurs d'ID</a>

À lire pour plus de détails : [www.yoyodesign.org/doc/w3c/css2/selector.html](http://www.yoyodesign.org/doc/w3c/css2/selector.html)

## Exemples sur les sélecteurs

```

<HTML>
  <HEAD>
    <TITLE>CSS - Appliquer un style (1)</TITLE>
    <STYLE type="text/css">
      * { color: #ff0000; }
      p { color: #0000ff; }
      p.vert { color: #008000; }
      p#gris { color: #7F7F7F; }
      /*#gris p { color: #7F7F7F; }*/
      h2 em { color: pink; }
      p>em { color: yellow; }
      img + p { color: khaki; }
      h3,h4,h5,h6 { color: brown; }
    </STYLE>
  </HEAD>
  <BODY>
    <A href="">Un lien en rouge (sélecteur universel *)</A>
    <P>Un paragraphe en bleu (sélecteur de type p)</P>
    <P class="vert">Un paragraphe en vert (sélecteur de classe)</P>
    <P id="gris">Un paragraphe en gris (sélecteur d'id)</P>
    <H2>Un titre de niveau 2 en rouge (sélecteur universel *) <EM>et en rose (sélecteur descendant)
      </EM></H2>
    <P>Ceci sera en bleu (sélecteur de type p) ...
      <EM>et ceci en jaune (sélecteur d'enfant)</EM>
    </P>
    <IMG SRC="">
    <P>Un paragraphe en kaki (sélecteur d'enfant adjacent)</P>
    <H3>Un titre de niveau 3 en marron (regroupement)</H3>
    <H4>Un titre de niveau 4 en marron (regroupement)</H4>
    <H5>Un titre de niveau 5 en marron (regroupement)</H5>
    <H6>Un titre de niveau 6 en marron (regroupement)</H6>
  </BODY>
</HTML>

```

Exemple 1 : appliquer un style

Un lien en rouge (sélecteur universel \*)

Un paragraphe en bleu (sélecteur de type p)

Un paragraphe en vert (sélecteur de classe)

Un paragraphe en gris (sélecteur d'id)

**Un titre de niveau 2 en rouge (sélecteur universel \*)** *et en rose (sélecteur descendant)*

Ceci sera en bleu (sélecteur de type p) ... *et ceci en jaune (sélecteur d'enfant)*

Un paragraphe en kaki (sélecteur d'enfant adjacent)

**Un titre de niveau 3 en marron (regroupement)**

**Un titre de niveau 4 en marron (regroupement)**

**Un titre de niveau 5 en marron (regroupement)**

**Un titre de niveau 6 en marron (regroupement)**



```
<HTML>
  <HEAD>
    <TITLE>CSS - Appliquer un style (2)</TITLE>
    <STYLE type="text/css">
      .magenta { color: magenta; }
      #navy { color: navy; }
    </STYLE>
  </HEAD>
  <BODY>
    <DIV class="magenta">
      <H1>Un titre de niveau 1 en magenta (éléments bornés par l'élément div)</H1>
      <P>Un paragraphe en magenta (éléments bornés par l'élément div)</P>
    </DIV>
    <DIV id="navy">
      <H1>Un titre de niveau 1 en navy (éléments bornés par l'élément div)</H1>
      <P>Un paragraphe en navy (éléments bornés par l'élément div)</P>
    </DIV>
    <P>
      <SPAN class="magenta">
        Ceci sera en magenta (élément span)
      </SPAN>
      ... et ceci sera en noir (par défaut) ...
    </P>
  </BODY>
</HTML>
```

*Exemple 2 : appliquer un style*

## Un titre de niveau 1 en magenta (éléments bornés par l'élément div)

Un paragraphe en magenta (éléments bornés par l'élément div)

## Un titre de niveau 1 en navy (éléments bornés par l'élément div)

Un paragraphe en navy (éléments bornés par l'élément div)

Ceci sera en magenta (élément span) ... et ceci sera en noir (par défaut) ...

## Principes techniques

### Cascade de CSS

Les feuilles de style ont trois origines différentes :

- L'**auteur** : produit des feuilles de style pour un document source en y étant incorporées ou reliées à celui-ci.
- L'**utilisateur** : peut être capable d'indiquer une information de style pour un document particulier.
- L'**agent utilisateur** (le plus souvent désigne le **navigateur**) : l'agent utilisateur conforme doit appliquer sa feuille de style par défaut avant toutes les autres feuilles de style d'un document.

La cascade de CSS définit un ordre de priorité, ou poids, pour chaque règle de style.

Les règles des feuilles de style de l'auteur ont, par défaut, plus de poids que celles de l'utilisateur. Au contraire, l'ordre de priorité est inversé pour les règles "**!important**". Les règles d'un auteur et d'un utilisateur sont prioritaires sur celles de la feuille de style par défaut de l'agent utilisateur (le navigateur).

### Ancêtres, Parents, Enfants et Frères

Chaque document HTML est toujours composé de conteneurs. Ceux-ci peuvent être ancêtres, parents, enfants ou frères :

- Un élément **Ancêtre** est un élément qui contient un élément ou une hiérarchie d'éléments.
- Un bloc **Parent** est un élément contenant directement un autre bloc. Par exemple, un **DIV** contenant un paragraphe **P**. Attention : si ce paragraphe contient lui-même des éléments (par exemple **STRONG**), **DIV** ne sera pas Parent de l'élément **STRONG** mais uniquement son Ancêtre. Le Parent est donc l'Ancêtre immédiat.
- Un bloc contenu directement dans un autre bloc est dit **Enfant** de cet élément. Par exemple, ici les éléments **LI** sont enfants de leur conteneur **UL**.
- Les éléments ayant le même élément Parent sont appelés **Frères**.

### Notion d'héritage

Les éléments enfants héritent de certaines valeurs de leurs éléments parents dans l'arbre du document. Chacune des propriété définit si elle est héritée, ou non.

Par exemple ici, tous les descendants de l'élément **BODY** auront la valeur de couleur '**black**', la propriété '**color**' étant héritée :

```
BODY { color: black; }
```



La valeur `inherit` provoque l'héritage des valeurs par les propriétés. Ceci s'applique également aux propriétés dont la valeur n'est normalement pas héritée.

## Unités de mesures

Les CSS proposent 8 unités possibles pour exprimer tailles et longueurs (sans compter les pourcentages). Le W3C les répartit en unités absolues et unités relatives :

- Les **unités absolues** sont : le point (**pt**), le pica (**pc**), le centimètre (**cm**), le millimètre (**mm**) et le pouce (**in**). Toutes ces unités sont équivalentes parce que proportionnelles entre elles (1 in = 2,54 cm = 25,4 mm = 72 pt = 6 pc). Le W3C précise que ces unités ne sont utiles que si les propriétés physiques du média de sortie sont connues et ce n'est jamais le cas pour les écrans. En pratique, ces unités ne sont utilisées que pour une impression sur papier.
- Les **unités relatives** sont : le « em » (**em**), le « ex » (**ex**), le pourcentage (%) et le pixel (**px**). Les deux premières unités sont relatives à la police de référence : 1 em est égal à la taille de cette police, tandis que 1 ex est la hauteur du « x » dans cette police (c.à.d. celle d'une lettre sans jambage). Le rapport em/ex dépend donc de la police utilisée. Le W3C parle d'unité « relative » pour dire que le résultat physique dépend du contexte.



Le W3C range le pixel parmi les unités relatives sous le prétexte qu'il n'a pas de taille bien définie puisqu'il varie d'une machine à une autre. Certains experts préfèrent parler d'unité « fixe » pour le pixel. Pour deux raisons : avec des tailles de texte en pixels, la taille du texte affiché ne tiendra pas compte des préférences de l'utilisateur, et le texte ne s'adaptera donc pas à la configuration et aux besoins de l'utilisateur. D'autre part, certains navigateurs ne permettent pas le redimensionnement « à la volée » d'un texte dimensionné en pixels.

Il convient d'ajouter que le W3C définit aussi des **tailles absolues** à l'aide des mots-clés suivants : **xx-small** (6,9 pt), **x-small** (8,3 pt), **small** (10 pt), **medium** (12 pt), **large** (14,4pt), **x-large** (17,28 pt) et **xx-large** (20,7 pt). Malheureusement, leur interprétation étant laissée aux soins du navigateur, on obtient des rendus différents en utilisant ce type de tailles.

## Flux normal

Le flux normal (ou courant) d'un document peut se définir comme étant le comportement naturel d'affichage des éléments d'une page web. Autrement dit, les éléments s'affichent dans l'ordre où ils sont déclarés dans le code HTML : verticalement, commençant en « haut » de l'écran pour aller jusqu'en « bas », et horizontalement de gauche à droite, sur la totalité de l'espace disponible et nécessaire en largeur comme en hauteur.

On distinguera deux groupes d'éléments :

- Les éléments de type **block** (**div**, **p**, **h1**, **h2**, **h3**, **h4**, **h5**, **h6**, **ul**, **ol**, **li**, **dl**, **dd**, **table**, **blockquote**, **pre**, **address**, etc.)
- Les éléments de type **inline** (**span**, **a**, **img**, **span**, **em**, **strong**, **cite**, **code**, **abbr**, etc.)

Un élément de type **bloc** (*block*) se différencie d'un élément de type **en-ligne** (*inline*) sur les principaux points suivants :

- Il occupe la totalité de la largeur de son conteneur
- Il permet l'attribution de marges verticales
- Il permet la modification de sa hauteur et largeur

Sauf exceptions, les éléments de type **en-ligne** n'occupent que la place minimum nécessaire à leur contenu.



De manière générale, les éléments de bloc peuvent contenir des éléments en-ligne et d'autres éléments de bloc. Et, les éléments en-ligne ne peuvent contenir que des données et d'autres éléments en-ligne.

```
<html>
  <head>
    <title>Boîtes de type bloc en flux normal</title>
    <style type="text/css">
      .rouge {background-color: #ff9999;}
      .vert {background-color: #99cc99;}
      .bleu {background-color: #9999cc;
width: 50%; height: 50px; margin-top: 50px; /* propriétés applicables pour un élément bloc */
      }
    </style>
  </head>
  <body>
    <p class="rouge">Une boîte rouge</p>
    <p class="vert">Une boîte verte</p>
    <p class="bleu">Une boîte bleue</p>
  </body>
</html>
```

*Exemple 1 : Boîtes de type bloc en flux normal*



```
<html>
  <head>
    <title>Boîtes de type en-ligne en flux normal</title>
    <style type="text/css">
      .rouge {background-color: #ff9999;}
      .vert {background-color: #99cc99;}
      .bleu {background-color: #9999cc;
width: 50%; height: 50px; margin-top: 50px; /* propriétés ignorées pour un élément inline */
      }
    </style>
  </head>
  <body>
    <p>
      <span class="rouge">Une boîte rouge</span>
      <span class="vert">Une boîte verte</span>
      <span class="bleu">Une boîte bleue</span>
    </p>
  </body>
</html>
```

Exemple 2 : Boîtes de type en-ligne en flux normal

Une boîte rouge Une boîte verte Une boîte bleue



On peut modifier ce comportement avec la propriété `display`.

```
<html>
  <head>
    <title>La propriété display</title>
    <style type="text/css">
      .rouge {background-color: #ff9999; display: block;}
      .vert {background-color: #99cc99; display: block;}
      .bleu {background-color: #9999cc; display: block;
width: 50%; height: 50px; margin-top: 50px; /* propriétés maintenant applicables (block) */
      }
    </style>
  </head>
  <body>
    <p>
      <span class="rouge">Une boîte rouge</span>
      <span class="vert">Une boîte verte</span>
      <span class="bleu">Une boîte bleue</span>
    </p>
  </body>
</html>
```

Exemple 3 : La propriété `display`

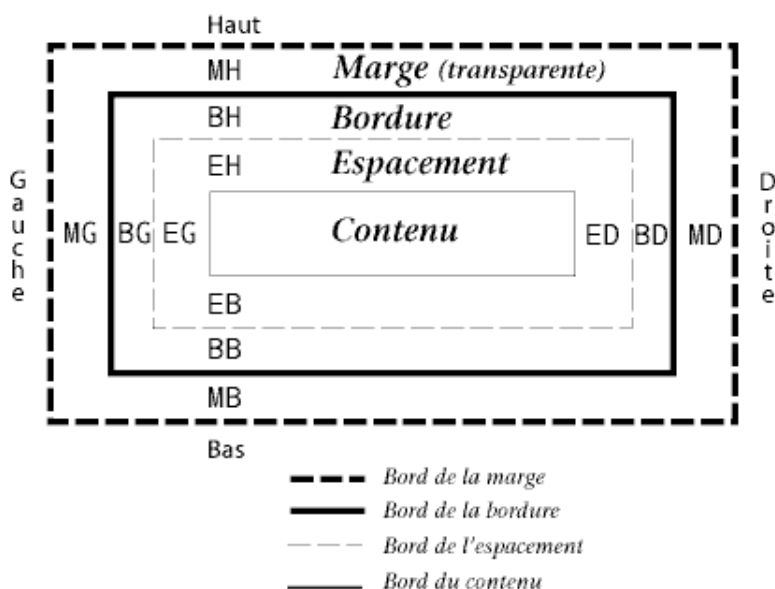
Une boîte rouge  
Une boîte verte

Une boîte bleue

À lire pour plus de détails : [Alsacrétions - Initiation au positionnement CSS](#)

## Les dimensions des boîtes

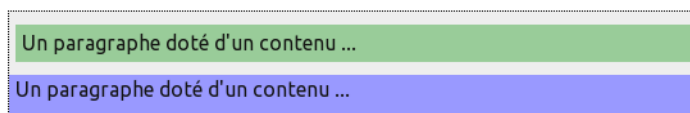
Chaque boîte possède une aire de contenu (un texte, une image, etc.) entourée en option par une aire d'espacement (*padding*), une aire de bordure (*border*) et une aire de marge (*margin*).



On peut travailler individuellement sur les propriétés *margin*, *padding* et *border* à l'aide des suffixes *-top* (haut), *-right* (droite), *-bottom* (bas), *-left* (gauche).

```
<html><head><title>Les marges intérieures et extérieures</title>
<style type="text/css">
.fond { background-color: #EEEEEE;
border: dotted 1px #000; /* fixe une bordure d'1 pixel en pointillé noir */
}
.vert { background-color: #99CC99;
margin: 10px 5px; /* fixe une marge de 10 pixels pour Haut/Bas et de 5 pixels pour Droite/Gauche */
padding: 5px; /* fixe un espacement de 5 pixels pour les quatre côtés */
}
.bleu { background-color: #9999FF;
margin: 0; /* fixe une marge nulle pour les quatre côtés */
padding: 2px 0px 15px 5px; /* fixe un espacement pour dans l'ordre : Haut, Droite, Bas, Gauche */
}
</style>
</head><body>
<div class="fond">
<p class="vert">Un paragraphe doté d'un contenu ... </p>
<p class="bleu">Un paragraphe doté d'un contenu ... </p>
</div>
</body></html>
```

*Exemple : Les marges intérieures et extérieures*



À lire pour plus de détails : [CSS2 - Le modèle des boîtes](#)

Les principales propriétés CSS à prendre en compte sont :

- la largeur : 'width'
- la hauteur : 'height'
- les marges : 'margin'
- l'espacement : 'padding'
- les bordures : 'border'
- la taille des caractères : 'font-size'
- l'interlignage : 'line-height'
- l'interlettrage (espacement entre les caractères du texte) 'letter-spacing'
- l'espace-mot : 'word-spacing'



Pour l'affichage des images dans leur taille nominale, les dimensions s'exprimeront alors en pixels.

Pour la **taille des caractères** (propriété CSS `font-size`), il sera conseillé d'utiliser :

- un pourcentage, pour signifier que le texte contenu dans cet élément aura une taille de x% de la taille du texte de son élément parent.
- une valeur en em, pour signifier que le texte contenu dans cet élément aura une taille de x fois la taille du texte de son élément parent.



En utilisant `font-size : Xem`, on ne demande pas une taille de texte fixe et absolue, mais une taille de texte proportionnelle à la taille de texte de l'élément parent. La valeur demandée, en em, sera un coefficient multiplicateur.

Un des effets de la cascade CSS : la taille des caractères avec des éléments div imbriqués !

```
<!DOCTYPE html PUBLIC "-//W3C//DTD html 4.0//EN">
<html>
  <head>
    <title>Les unités</title>
    <style type="text/css">
      div { font-size: 1.2em; }
    </style>
  </head>
  <body>
    <h1>L'effet de cascade</h1>
    <div>Du texte avec font-size: 1.2em<div>Du texte avec font-size: 1.2em<div>Du texte avec font-
      size: 1.2em<div>Du texte avec font-size: 1.2em</div></div></div></div>
  </body>
</html>
```

*Exemple : L'effet de cascade*

## L'effet de cascade

Du texte avec font-size: 1.2em

Du texte avec font-size: 1.2em

Du texte avec font-size: 1.2em

Du texte avec font-size: 1.2em

Concernant la **taille des blocs**, les utilisations les plus courantes sont :

- en pixels, pour éviter qu'un affichage s'étende trop en largeur en fonction de la résolution de l'écran.
- en pourcentages, très pratique pour se référer aux dimensions de la fenêtre ou du bloc contenant.
- en em, par exemple pour fixer (à peu près) le nombre de caractères par ligne.



Attention à la spécification de hauteur (`height`) des blocs car il y a un fort risque de débordement (cf. la propriété `overflow`). Le plus sage consiste à ne rien spécifier du tout, ou bien à demander une option « automatique » (`height : auto`), et le bloc s'ajustera tout seul en hauteur.

## Position

La propriété `position` permet de gérer les positions lorsque les possibilités du flux normal ne suffisent plus.

Le **positionnement relatif** (`position : relative`) permet d'inscrire un contenu en flux normal, puis de le décaler horizontalement et/ou verticalement. Le contenu précédent et suivant n'est pas affecté par ce déplacement, ce qui peut donc entraîner des chevauchements.

```
<html>
  <head>
    <title>Position relative</title>
    <style type="text/css">
      .jaune { position: relative; bottom: 5px; background-color: yellow; }
      .yellow { position: relative; bottom: 5px; left: 3em; background-color: #ffff00; }
    </style>
  </head>
  <body>
    <p>Du texte<span class="jaune">boîte en position relative</span>dans un paragraphe.</p>
    <p>Du texte<span class="yellow">boîte en position relative</span>dans un paragraphe.</p>
  </body>
</html>
```

*Exemple : La position relative*

Du texte **boîte en position relative** dans un paragraphe.

Du texte **boîte en position relative** dans un paragraphe.

Le **positionnement absolu** (`position : absolute`) « retire » totalement du flux le contenu concerné : sa position est maintenant déterminée par référence aux limites du conteneur (c'est à dire le plus souvent la fenêtre du navigateur). Un élément bénéficiant d'une position absolue ne bougera pas de sa position initiale tant que l'une des propriétés `top`, `bottom`, `left` ou `right` n'a pas été précisée ; il s'agit d'ailleurs là d'un comportement applicable à toutes les positions.



Un élément positionné en absolu se réfère non pas à son parent direct, mais au premier ancêtre positionné qu'il rencontre. L'élément, n'étant plus dans le flux normal, perd une de ses caractéristiques majeures qui est celle de recouvrir la totalité de la largeur disponible de l'élément parent.

```
<html><head><title>Position absolue</title>
  <style type="text/css">
    p { margin: 0; padding: 10px 5px; border: solid 1px #000;}
    .rouge { background-color: #FF9999; position: absolute; top: 1%; left: 1%; width: 20%; }
    .verte { background-color: #99CC99; position: absolute; top: 1%; right: 1%; width: 20%; }
```

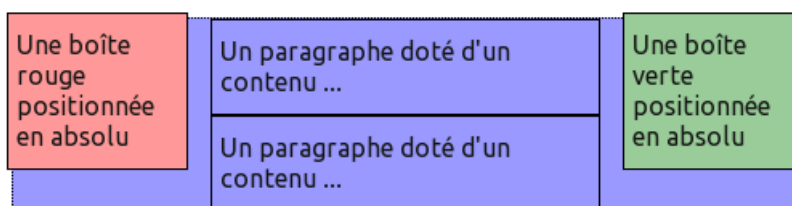


```

.bleue { background-color: #9999FF; padding: 0 25%; border: dotted 1px #000;}
</style>
</head>
<body>
  <div class="bleue">
    <p>Un paragraphe doté d'un contenu ... </p><p>Un paragraphe doté d'un contenu ... </p>
    <p class="rouge">Une boîte rouge positionnée en absolu</p>
    <p class="verte">Une boîte verte positionnée en absolu</p>
  </div>
</body>
</html>

```

*Exemple : La position absolue*



Le **positionnement fixe** (`position : fixed`) s'apparente au positionnement absolu, à l'exception des points suivants :

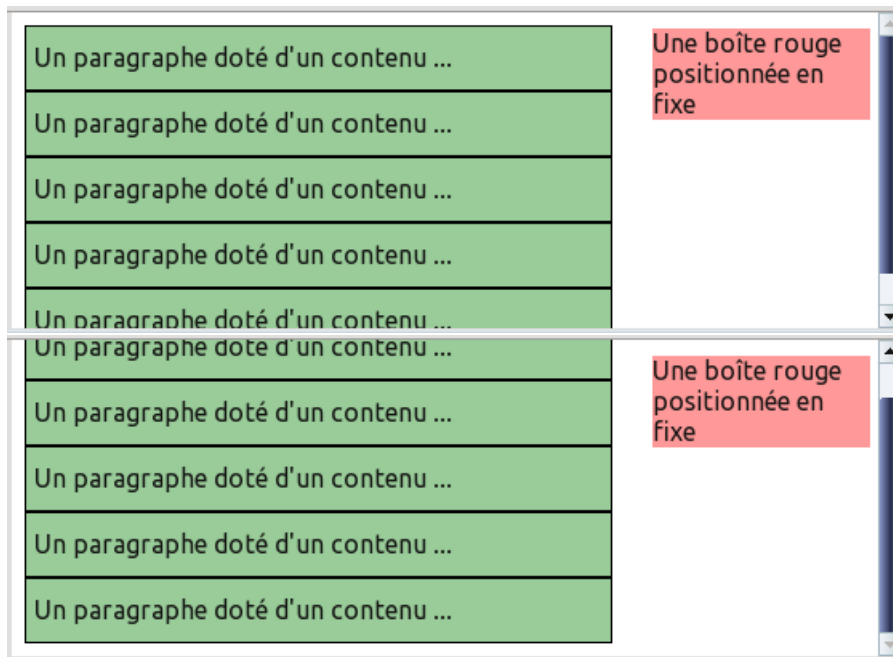
- Lorsque le positionnement est précisé (`top`, `bottom`, `left` ou `right`), l'élément est toujours positionné par rapport à la fenêtre du navigateur
- L'élément est fixé à un endroit et il ne bouge plus de la position initialement définie, même lors de la présence d'une barre de défilement.

```

<html>
  <head>
    <title>Position fixe</title>
    <style type="text/css">
      p { margin: 0; padding: 10px 5px; border: solid 1px #000;}
      .rouge { background-color: #FF9999; position: fixed; top: 5%; right: 1%; width: 25%;}
      .vert { background-color: #99CC99; margin: 0 0; width: 70%;}
    </style>
  </head>
  <body>
    <div class="rouge">
      Une boîte rouge positionnée en fixe
    </div>
    <div class="vert">
      <p>Un paragraphe doté d'un contenu ... </p><p>Un paragraphe doté d'un contenu ... </p>
      <p>Un paragraphe doté d'un contenu ... </p><p>Un paragraphe doté d'un contenu ... </p>
      <p>Un paragraphe doté d'un contenu ... </p>
    </div>
  </body>
</html>

```

*Exemple : La position fixe*



Les blocs positionnés en "absolu" ou "fixé" sortent du flux naturel et ils ne sont alors plus dépendant des éléments frères. Pour se placer, un tel bloc se réfère non pas à son parent direct, mais au premier ancêtre positionné qu'il rencontre.

La **position statique** (`position : static`) correspond simplement à la valeur par défaut d'un élément du flux. Il n'y a que peu d'intérêt à la préciser, si ce n'est dans le but de rétablir dans le flux un élément qui serait positionnée hors du flux.

## Boîte flottante

Une boîte flottante est retirée du flux normal, et placée le plus à droite (`float : right`) ou le plus à gauche (`float : left`) possible dans son conteneur. Le contenu suivant cette boîte flottante s'écoule le long de celle-ci dans l'espace laissé libre. On peut empêcher ce comportement avec la propriété `clear`.

```
<html>
  <head>
    <title>Boîtes flottantes</title>
    <style type="text/css">
      .jaune { background-color: yellow; float: right; width: 10%; margin: 0; }
      .rouge { background-color: #ff9999; float: left; width: 10%; margin: 0; }
      .verte { background-color: #99cc99; }
      .bleue { background-color: #9999cc; }
    </style>
  </head>
  <body>
    ... <!-- voir exemples ci-dessous -->
  </body>
</html>
```

*Exemple : Les règles de style*

```
<p class="jaune">Une boîte jaune flottant à droite</p>
<p class="verte">Une boîte verte dotée d'un contenu plus long ... plus long... plus long... plus
long... plus long... plus long... plus long... plus long... plus long... plus long... plus long
... plus long... plus long... plus long... plus long... plus long... plus long... plus long...
```





## Liens

Ce document a été rédigé à partir de la documentation CSS2 du W3C et d'articles publiés sur les sites : [www.alsacreations.com](http://www.alsacreations.com) et [openweb.eu.org](http://openweb.eu.org).

Pour aller plus loin, il est vivement conseillé de lire :

- [Alsacréations - Polices, quelle taille choisir ?](#)
- [Alsacréations - Typographie web](#)
- [Openweb - Initiation au positionnement CSS : 1.flux et position relative](#)
- [Openweb - Initiation au positionnement CSS : 2.position float](#)
- [Openweb - Initiation au positionnement CSS : 3. position absolue et fixe](#)
- [Alsacréations - Initiation au positionnement en CSS \(Partie 1\)](#)
- [Alsacréations - Initiation au positionnement en CSS \(Partie 2\)](#)
- [Alsacréations - Comprendre la structure HTML et le rendu CSS des balises : bloc et en-ligne](#)
- [Alsacréations - Comment positionner les éléments en CSS ?](#)
- [Alsacréations - Guide de survie du positionnement CSS](#)
- [Alsacréations - Centrer les éléments ou un site web en CSS](#)
- [Openweb - Initiation au centrage CSS](#)
- [Alsacréations - Le contexte de formatage block en CSS](#)
- [Openweb - Cascade CSS et priorité des sélecteurs](#)
- [Alsacréations - Gabarits HTML et CSS simples](#)