Comprendre et Améliorer une Feuille CSS

Thomas ZILLIOX

Juillet 2011

1. Element block

1. Le Box model
2. La largeur
3. La hauteur
4. Les Débordements
5. Les marges
6. La position verticale
7. La position horizontale

2. Element inline

1. Les dimensions
2. La position horizontale
3. La position verticale

3. Mise en page

1. Les éléments float
2. La position relative
3. La position absolute

4. Mise en forme

1. Le gabarit
2. Le texte
3. Les unités

5. La Feuille de style

1. Les poids des sélecteurs
2. Les opérateurs
3. L'organisation

6. Un peu d'HTML

1. HTML valide
2. Les média queries
3. Les commentaires conditionnels

7. Les outils

1. Firebug

Pourquoi le CSS est-il intéressant ?

Crucial : C’ est la techno qui va décider de la qualité du design d’un site web

Savoir-faire : Le rendu doit être bon sur les différents navigateurs, écran, OS, etc.

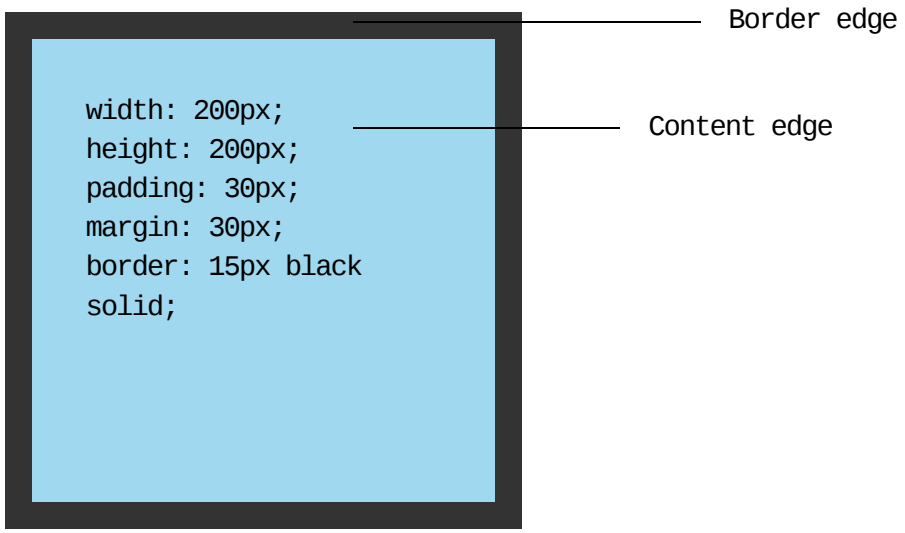
Sincère : Ouvert par nature, vous pouvez étudier le CSS de Yahoo, Google, etc.

Introduction

Element block

Element block > Le Box model

Le Box model est la première chose à comprendre en CSS.



Nous demandons en CSS que cet élément ait une largeur de **200px**.

Pourtant, il prend une place de **350px** sur notre écran.

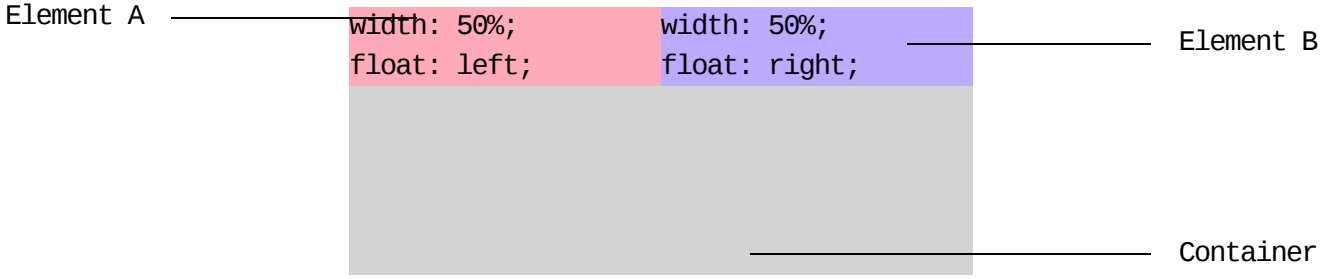
Element block > Le Box model

Le Box model peut surprendre.

Nous avons deux éléments qui sont l'un à côté de l'autre.

Chacun occupe la moitié (**50%**) de la largeur de leur parent.

Nous voulons ajouter une bordure sur le second élément pour les séparer.



La largeur totale utilisée sera alors de **50% + 50% + 1px**.

Les deux éléments ne pourront plus tenir l'un à côté de l'autre.

En CSS2, on ne peut pas régler ce problème sans ajouter un **div** supplémentaire.

En CSS3, on peut demander au deuxième élément de mesurer **calc(50% – 1px)**.

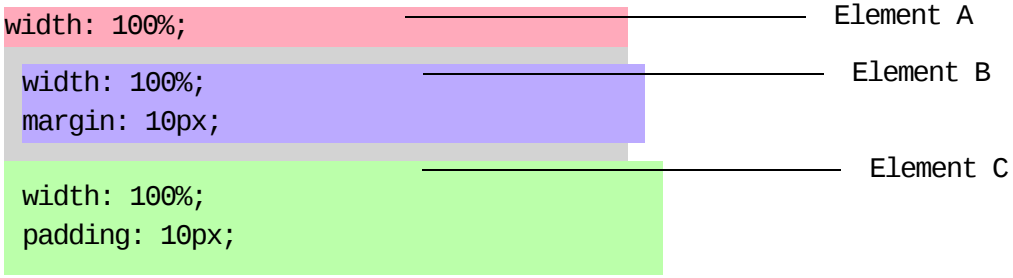
Element block > La largeur

Les éléments suivants sont des élements block :

** <h1>…<h6> <p>   <table> <blockquote> <pre> <form>**

On peut spécifier leur largeur en **px**, ou en **%** de la largeur de leur élément parent.

Si la largeur dépasse celle de son parent, on sera en présence d'un débordement :

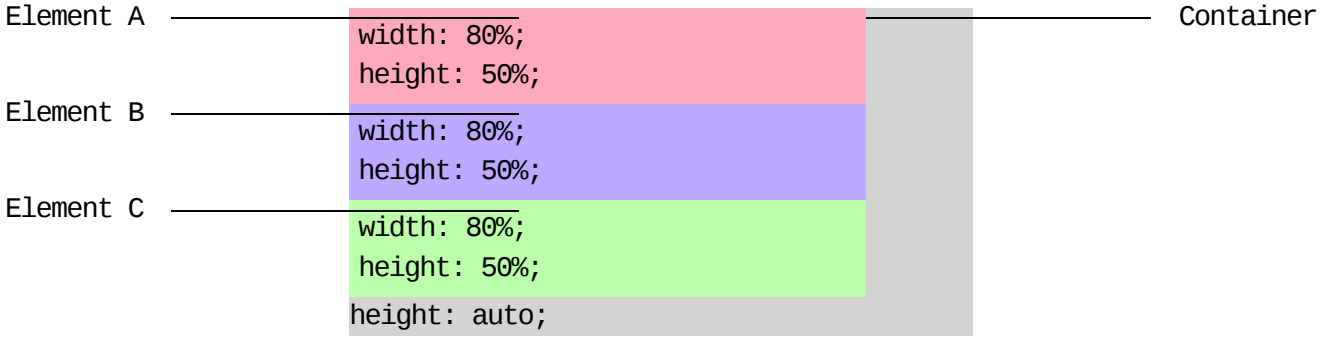


Avec la valeur par défaut, **width: auto**, la largeur de l'élément block s'adapte pour être le plus grand possible tout en évitant les débordements.

Il est donc conseillé de ne pas spécifier inutilement la largeur d'un élément block.

Element block > La hauteur

Avec la valeur par défaut, **height: auto**, la hauteur d'un élément block s'adapte à son contenu. Les hauteurs des fils exprimées en **%** ne seront pas pris en compte.



Si l'on spécifie une hauteur à un élément block, celle-ci ne s'adaptera plus.

Les fils risquent alors de déborder du parent.

Il est donc conseillé de ne pas spécifier inutilement la hauteur d'un élément block.

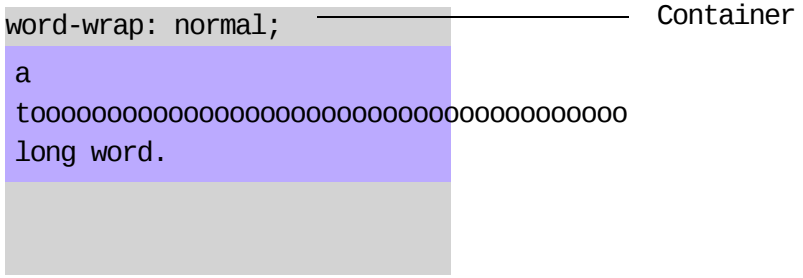
Par contre, les hauteurs des fils exprimées en **%** seront pris en compte.

Element block > Les Débordements

Il faut faire attention au renvoie à la ligne spécifié avec **word-wrap**.

La valeur par défaut **normal** peut causer des débordements.

La valeur **break-word** permet de forcer la césure d'un mot trop long.



Il est possible de gérer les débordements avec la propriété **overflow**.

Les valeurs possibles sont **visible** valeur par défaut, **hidden** et **scroll**.

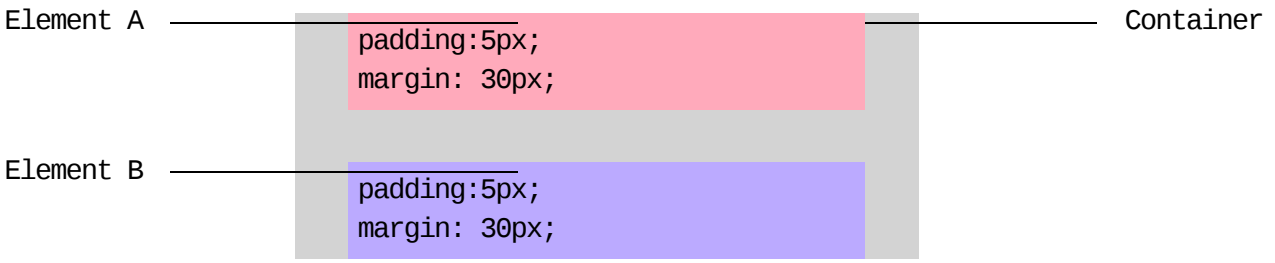
On peut utiliser les propriétés détaillées **overflow-x** et **overflow-y**.

Element block > Les marges

margin et **padding** ne fonctionnent pas de manière équivalente :

Les **margin** verticales ne s'appliquent pas avec un parent qui n'a pas de contenu textuel, **border** ou **padding**

Les **margin** entre éléments frères se superposent



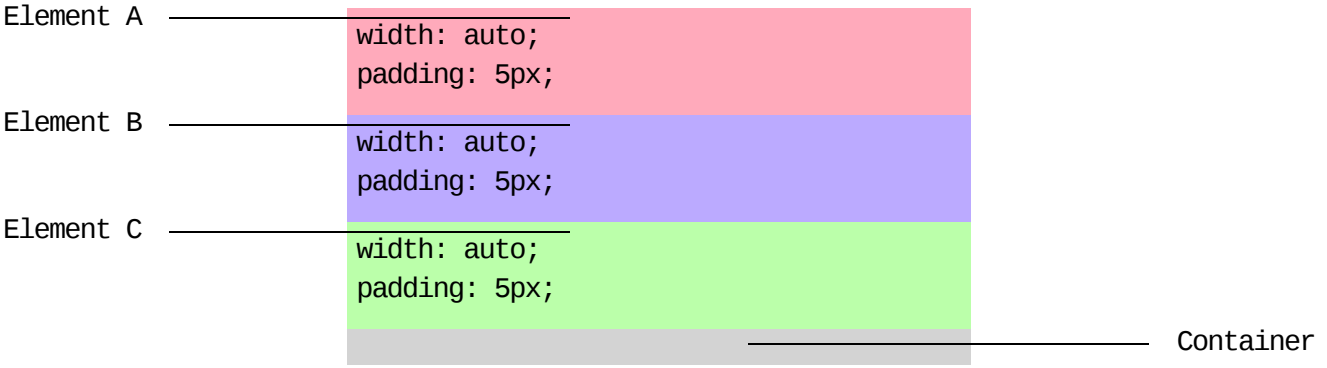
Si vous avez le choix, préférez utiliser la propriété **padding** que la propriété **margin**.

Le rendu du site résistera plus aux futures modifications.

Element block > La position verticale

Un élément block se positionne en fonction de la position de son frère ainé.

Les éléments blocks s'empilent de haut en bas, on parle de **flux vertical**.



La position d'un élément block ne dépend que de la hauteur de ses frères ainés.

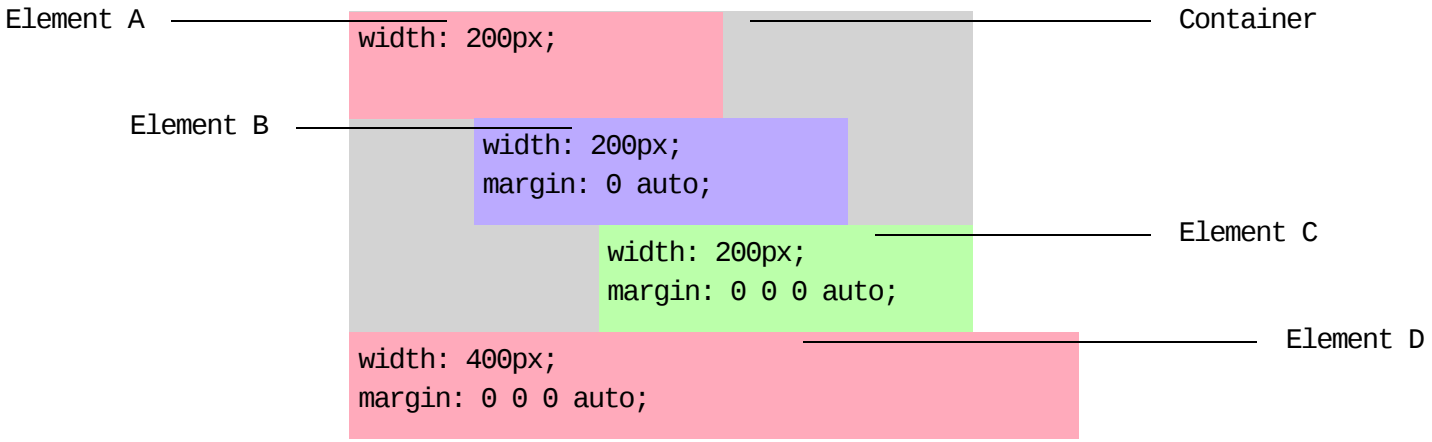
Pour désigner l'élément invisible qui réserve toute la largeur de l'élément parent, on parle d'une **empreinte de flux**.

Element block > La position horizontale

On peut positionner verticalement un élément block dans un autre élément block.

On peut indiquer à l'élément fils d'être centré, aligné à gauche ou à droite.

Pour cela on utilise la propriété **margin** sur l'élément fils.



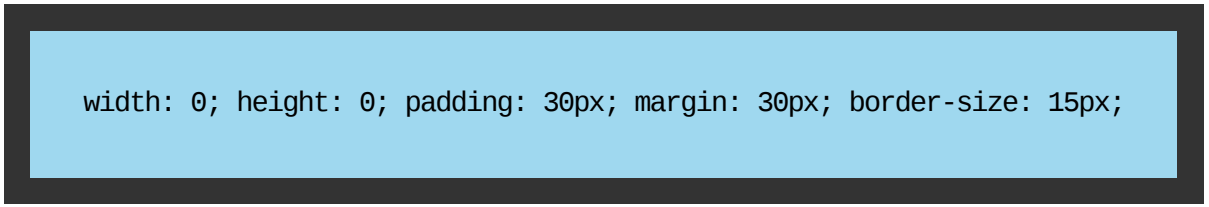
Ceci ne fonctionne que si l'élément fils est plus petit que l'élément parent.

Si le sens d'écriture est de gauche à droite , **direction : ltr** , alors le débordement se fera toujours à droite.

Element inline

Element inline > Les dimensions

Les éléments inline n'ont pas le même box model que les éléments block.



La propriété **width** n'est pas interprétée. La largeur s'adapte à son contenu.

La propriété **height** n'est pas interprétée. La hauteur dépend de la taille d'écriture, donc de la propriété **font-size**.

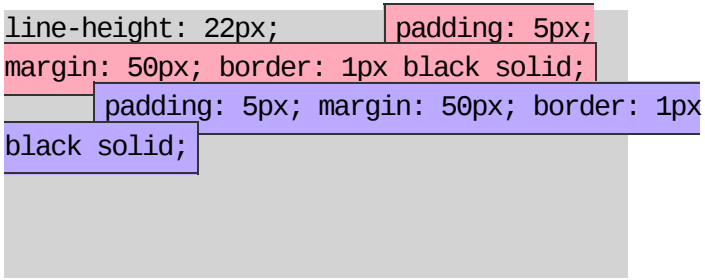
Les propirétés **margin-top** et **margin-bottom** n'ont pas d'effet.

Element inline > Les dimensions

Le boxmodel des éléments inline , qui sont sur plusieurs lignes , diffèrent beaucoup de celui des éléments block.

padding-left , **margin-left** et **border-left** ne sont appliqué qu'à la première ligne.

padding-right , **margin-right** et **border-right** ne sont appliqué qu'à la dernière ligne.



padding-top , **padding-bottom** , **border-top** et **border-bottom** ne modifient pas l'interligne. Il faut le modifier à l'aide de la propriété **line-height** sur son parent.

Element inline > La position horizontale

Un élément inline se positionne en fonction de la largeur de ses frères.

Les éléments inline s'empilent les uns à côté des autres, on parle de **flux horizontal**.

```
display: block; text-align: left;
display:inline display:inline display:inline
```

```
display: block; text-align: center;
display:inline display:inline display:inline
```

```
display: block; text-align: right;
display:inline display:inline display:inline
```

la position horizontale des éléments inline contenu dans un élément block dépend de la propriété **text-align** du parent.

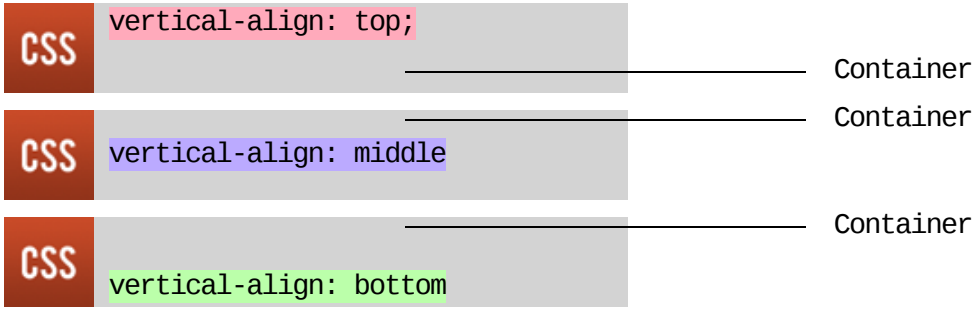
La valeur par défaut est **inherit** et **left** sur le body.

Element inline > La position verticale

On peut également contrôler l'alignement vertical des éléments inline.

Pour ceci, on utilise la propriété **vertical-align** sur les éléments inline.

Les valeurs courament utilisés sont **top middle** et **bottom**.



vertical-align fonctionne très bien entre une image et du texte.

Il faut juste appliquer la même valeur au texte et à l'image.

Il ne faut donc pas hésiter à l'utiliser.

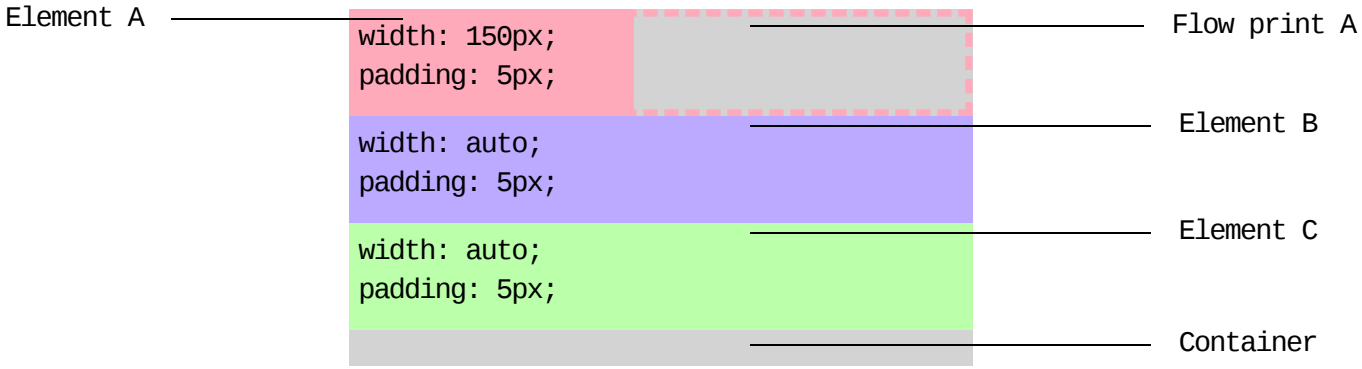
Mise en page

Mise en page > Les éléments float

Un élément peut être **float: none** , valeur par défaut , **left** ou **right**.

Un élément flottant s'aligne le bord gauche ou droit de son parent.

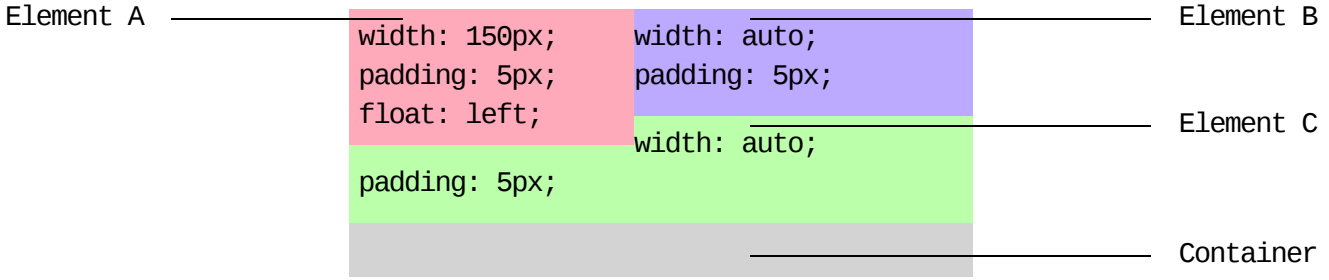
Un élement flottant n'a pas d'emprunte dans le flux. Les éléments non-flottants se positionneront donc sans prendre en compte les éléments flottants.



Les éléments flottants ne passent pas au-dessus des contenus textuels.

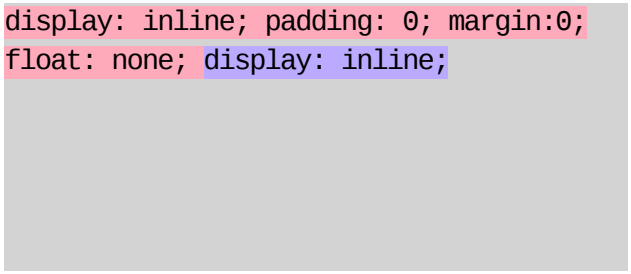
Mise en page > Les éléments float

Si on ne spécifie pas de largeur à un élément flottant, elle s'adapte à son contenu.



Un élément flottant est toujours un élément block.

Mettre un élément en flottant peut donc nous empêcher l'alignement horizontal :



Mise en page > Les éléments float

Si on ne spécifie pas de hauteur à un élément bloc, elle s'adapte à son contenu.

Cette hauteur est en fait la somme des hauteurs des empreintes de ses fils.

Si un élément bloc ne contient que des éléments flottants, il aura un hauteur nulle.

Il est possible de nettoyer le flux avec la propriété **clear**.

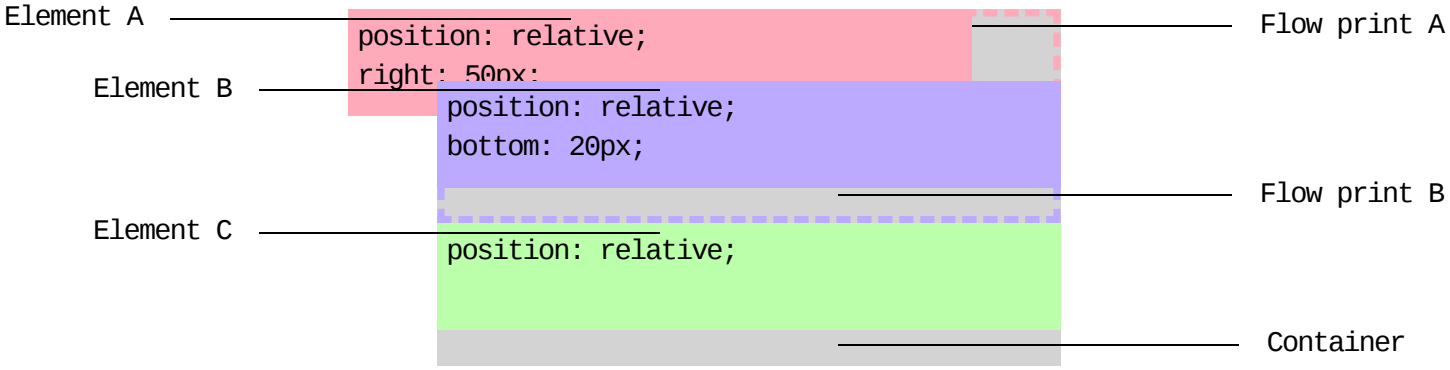
Les valeurs possibles sont **none**, valeur par défaut , **left**, **right** et **both**.

Si un élément est **clear: left**, il se placera en-dessous des éléments **float: left**

On peut également nettoyer le flux sans ajouter d'élément à l'aide d'un clearFix.

Mise en page > La position relative

Avec la position relative, on peut déplacer un élément relativement à son empreinte dans le flux. L'empreinte n'est par contre pas modifiée.



Il peut y avoir superposition, on gère les conflits de profondeur avec **z-index**.

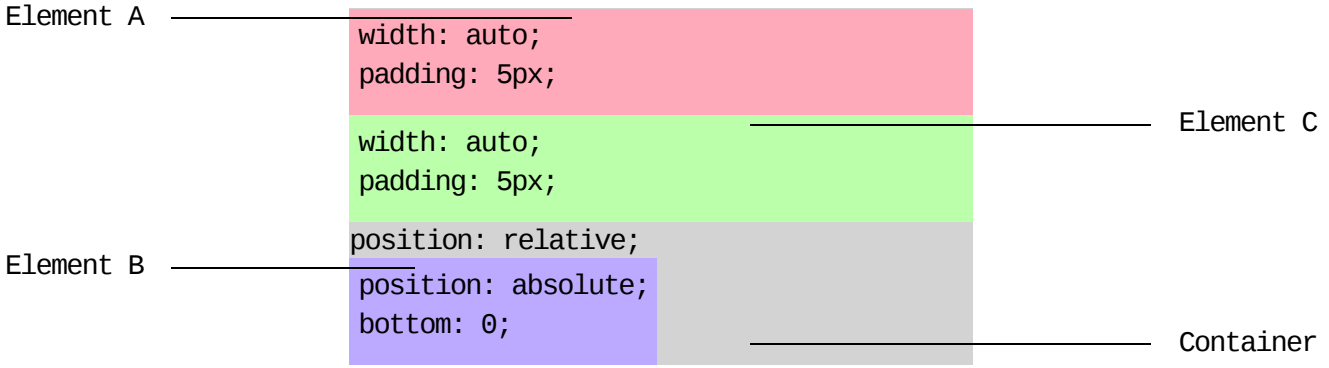
La valeur par défaut est **0**. Celui qui a la plus grande valeur passe au-dessus.

Dans le cas d'un **z-index** équivalent, l'élément le plus bas dans la page HTML passe au-dessus.

Mise en page > La position absolute

Avec la position absolute, on peut déplacer et dimensioner un élément en fonction de son premier parent positionné. Les dimensions en **%** sont en fonction du premier parent positionné, même si sa hauteur n'est pas explicite.

Il n'a pas d'empreinte dans le flux et sa largeur s'adapte à son contenu.



Par défaut, l'élément en position absolute se place là où il aurait du être.

Comme pour les position relative, on gère la profondeur avec **z-index**.

Un élément en position absolute est toujours block.

Mise en forme

Mise en forme > Le gabarit

Quelques propriétés utiles pour la mise en forme des éléments :

```
#header {
  background-color: white;
  background-image: url(foo.png)
  background-repeat: repeat-x;
  background-attachment: fixed;
  background-position: top right;
  border-size: 10px;
  border-style: solid;
  border-color; #77ccff;
  border-radius: 5px;
  opacity: 0.75;
}
```

Mise en forme > Le texte

Quelques propriétés utiles pour la mise en forme du texte :

```
h3 {
  color: #333333;
  font-style: italic;
  font-variant: small-caps;
  font-weight: bold;
  font-size: 30px;
  font-family: “Open Sans”,Arial,sans-serif;
  letter-spacing: -1px;
  line-height: 36px;
  text-align: left;
  text-decoration: underline;
  text-indent: 1em;
  text-transform: uppercase;
  word-spacing: 5px;
  word-wrap: break-word;
}
```

Mise en forme > Les unités

Les unités sont généralement source d'erreurs.

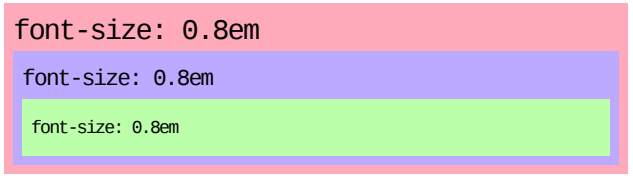
Utilisez toujours des **px** ou des **%** pour les dimensions, évitez les :

ex ou **em** : La valeur dépend du font-size

cm , **mm** , **in** , **pc** ou **pt** : La valeur dépend de l'écran

Utilisez toujours des **px** pour **font-size**, évitez les :

ex , **em** ou **%** : La valeur dépend du font-size du parent



La Feuille de style

La Feuille de style > Les poids des sélecteurs

```
<div id="article">
  <p>Text 1</p>
  <p class="text">Text 2</p>
  <p class="text last">Text 3</p>
</div>
```

```
p { margin: 0; padding: 5px;}
```

Text 1
margin: 0; padding: 5px;

Text 2
margin: 0; padding: 5px;

Text 3
margin: 0; padding: 5px;

La Feuille de style > Les poids des sélecteurs

```
<div id="article">
  <p>Text 1</p>
  <p class="text">Text 2</p>
  <p class="text last">Text 3</p>
</div>
```

```
p { margin: 0; padding: 5px;}
.text.last{padding-left:60px}
```

Text 1
margin: 0; padding: 5px;

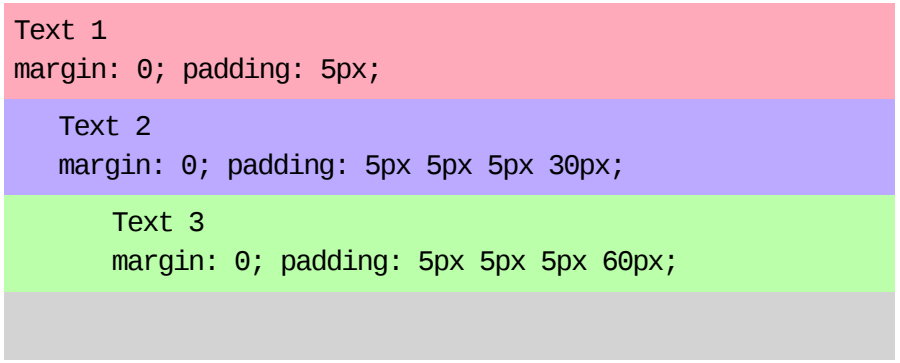
Text 2
margin: 0; padding: 5px;

Text 3
margin: 0; padding: 5px 5px 5px 60px;

La Feuille de style > Les poids des sélecteurs

```
<div id="article">
  <p>Text 1</p>
  <p class="text">Text 2</p>
  <p class="text last">Text 3</p>
</div>
```

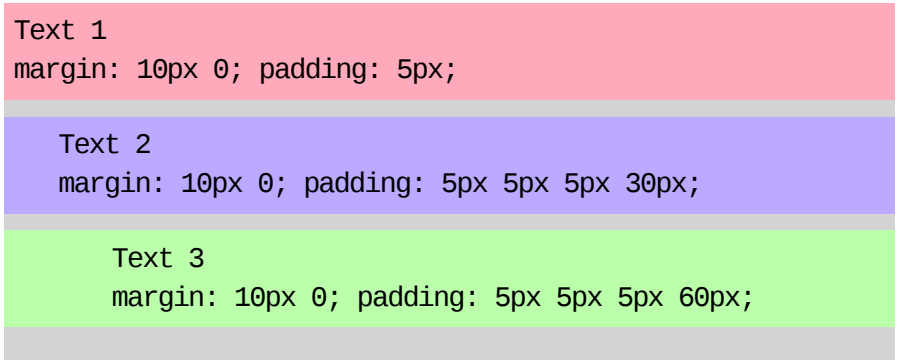
```
p { margin: 0; padding: 5px;}
.text.last{padding-left:60px}
.text { padding-left: 30px; }
```



La Feuille de style > Les poids des sélecteurs

```
<div id=“article”>
  <p>Text 1</p>
  <p class=“text”>Text 2</p>
  <p class=“text last”>Text 3</p>
</div>
```

```
p { margin: 0; padding: 5px;}
.text.last{padding-left:60px}
.text { padding-left: 30px; }
p { margin: 10px 0; }
```



La Feuille de style > Les poids des sélecteurs

```
<div id=“article”>
  <p>Text 1</p>
  <p class=“text”>Text 2</p>
  <p class=“text last”>Text 3</p>
</div>
```

```
p { margin: 0; padding: 5px;}
.text.last { padding-left: 60px;}
.text { padding-left: 30px;}
p { margin: 10px 0;}
‡article p { padding-left: 10px;}
```

Text 1
margin: 10px 0; padding: 5px 5px 5px 10px;

Text 2
margin: 10px 0; padding: 5px 5px 5px 10px;

Text 3
margin: 10px 0; padding: 5px 5px 5px 10px;

La Feuille de style > Les poids des sélecteurs

Voici les règles de priorités en CSS :

1. Quelles propriétés sont marquées comme !important
2. Quelles propriétés proviennent de l' attribut style
3. Quelles propriétés ont le sélecteur le plus lourd :
    1. Plus de sélecteurs d' id **#popup_login**
    2. Plus de sélecteurs de classe **.lien_ajax** , **: hover** , **[href="#"]**
    3. Plus de sélecteurs d' élement **a** , **:: first-letter**
    4. Plus de sélecteur universel *****
4. Laquelle est chargée en dernier
5. Valeurs par défaut du navigateur, parfois **inherit**

La Feuille de style > Les opérateurs

Les sélecteurs simples peuvent être combinés avec des opérateurs :

p.last : Eléments qui sont <p> et class="last"

div .last : Eléments class="last" qui ont un parent

div > .last : Eléments class="last" dont le parent est

p + .last : Eléments class="last" qui suivent un <p>

A noter sur les opérateurs :

Les opérateurs ne rajoutent pas de poids à un sélecteur

Il n' existe pas de sélecteurs de parents ou de précédents

La Feuille de style > L'organisation

Une feuille de style peut devenir grande, il faut l'organiser :

1. Normalize CSS (reset déconseillé)
2. Déclarations universelles : body , p , a
3. Bibliothèque : .photo , .important
4. Styles du gabarit : #sidebar , #header
5. Styles du contenu
6. Styles du contenu spécifique à une page

Garder un style d'écriture constant :

```
.important,
.very_important {
   font-weight: bold;
   margin-top: 1em;
}
```

La Feuille de style > Quelques conseils

Le CSS est sensible à la casse

Pour avoir une CSS réutilisable :

Eviter Les propriétés marquées comme !important

Eviter le style dans l’atribut style

Eviter d’indiquer inutilement la balise

Avoir des noms de classe / id sémantiques

Avoir une convention de nommage

Pour garder votre design flexible :

Ne pas abuser des positions absolute et relative

Rétablir le flux après les éléments flottants

Un peu d'HTML

Un peu d'HTML > HTML valide

Il est important d'écrire de l'HTML un minimum valide.

Sinon notre CSS peut nous surprendre.

Voici quelques points d'attention :

Ne pas oublier le DOCTYPE

Règles de descendance :

<p> ne peut pas contenir d'élement Block

Les éléments inline ne peuventt contenir d'éléments Block

<ul> **<ol>** ne peuvent contenir que des **<li>**

<table> > [**<tbody>** >] **<tr>** > **<td>** à respecter

Un peu d'HTML > Les média queries

Les médias CSS permettent de spécifier les styles à appliquer en fonction du media. On peut modifier les règles CSS :

Pour l'impression, la projection ou les télévisions

En fonction de la taille de l'écran

Pour la visualisation en portait ou en paysage

```
<link rel="stylesheet" type="text/css" media="all" href="style.css" />
<link rel="stylesheet" type="text/css" media="print" href="print.css" />
<link rel="stylesheet" media="screen and (max-width: 640px)" href="smallscreen.css" />
<link rel="stylesheet" type="text/css" media="all and (orientation:portrait)" href="portrait.css" />
```

Une feuille de style d'impression est souvent rapide à faire, pensez-y !

Un peu d'HTML > Les commentaires conditionnels

Les commentaires conditionnels permettent les feuilles de styles spécifiques :

à un navigateur

à une ou plusieurs version d'un navigateur

```
<!–[if IE]> <link type="text/css" rel="stylesheet" href="styles-ie.css" /> <![endif]–>
<!–[if IE 6]> <link type="text/css" rel="stylesheet" href="styles-ie6.css" /> <![endif]–>
<!–[if lte IE 8]> <link type="text/css" rel="stylesheet" href="styles-old-ie.css" /> <![endif]–>
```

Pensez-y pour corriger l'apparence de votre site sous IE6 :

Ils ne perturbent pas l'apparence sur les autres navigateurs

Ils permettent de regrouper toutes les modifications spécifiques

Les outils

Les outils > Firebug

Firebug est une extension Firefox.

Il permet de modifier son CSS en direct et voir pour un élément donné :

Ses propriétés par poids de sélecteurs

Les propriétés héritées

Les styles hover

Le box model d'un élément

D'autres outils peuvent être pratiques, notamment :

Validez ses feuilles CSS

Pensez aux autres résolutions

Quelques conseils (ne pas tout prendre au pied de la lettre)

Trouver les selecteurs inutilisés

Cette présentation est faite de HTML5 / CSS3 / JS / PHP

Elle utilise de nombreux projets open source :

Html5Slides : template des slides (Apache2.0)

RainTPL : moteur de template (LGPL 3)

Textile : moteur de markup (Modified BSD)

Elle a été rédigée par Thomas ZILLIOX d' OPEN WIDE

Merci pour votre attention