

Secteur Tertiaire Informatique  
Filière étude - développement.

Développer des composants d'interface

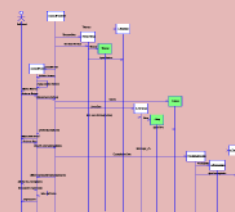
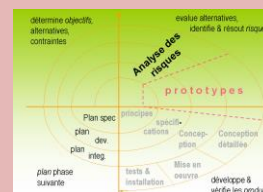
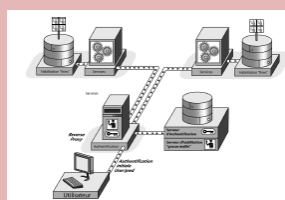
## Initiation à CSS3

Accueil

Apprentissage

Période en  
entreprise

Evaluation



CODE BARRE

# SOMMAIRE

Les feuilles de styles CSS.....	3
I Objectifs des feuilles de style.....	3
II Utiliser les feuilles de styles en HTML.....	4
III Définition des formats (sélecteurs) .....	5
IV Les balises <div> et <span>... et les autres... ..	7
V Le modèle de « boîtes ».....	8
VI Définition des propriétés.....	8
VII Le positionnement des éléments .....	11
VII.1 Le positionnement en flux normal .....	14
Boîte de type bloc en flux normal.....	14
Boîte de type en ligne en flux normal.....	14
VII.2 Le positionnement en flux relatif .....	15
VII.3 Le positionnement flottant.....	16
Les problèmes de débordement : .....	17
VII.4 Le positionnement absolu ou fixe.....	19
Profondeur des éléments.....	20
VIII Les unités de mesure .....	21
IX Les pseudo-attributs.....	21
X Les ombrages .....	22
XI Les angles arrondis.....	23
XII Les fonds dégradés .....	23
XIII La transparence.....	24
XIV Les transformations, transitions et animations .....	25
XV Les jeux de styles différents selon le périphérique.....	26

# LES FEUILLES DE STYLES CSS

## I OBJECTIFS DES FEUILLES DE STYLE

Les feuilles de style (CSS : **C**ascading **S**tyle **S**heet) complètent HTML, en offrant un langage permettant de **définir des propriétés de format et de présentation au contenu des éléments définis par des balises HTML**. On peut par exemple affecter à certains éléments une couleur d'arrière plan, une image d'arrière plan, des bordures, des polices de caractères spécifiques, etc...

CSS permet d'aller plus loin, beaucoup plus loin, que HTML dans la présentation des pages Web ; on peut ainsi par exemple positionner au pixel près certains éléments dans la fenêtre d'affichage du navigateur, ou modifier l'aspect du curseur ou encore effectuer certaines animations, le tout, sans aucune programmation, rien qu'avec des listes de valeurs d'attributs.

Les styles peuvent s'appliquer balise par balise, ou bien de manière commune à toutes (ou certaines) balises d'un type donné. Ceci permet de **définir un style-type** une seule fois, et de **l'appliquer autant de fois que nécessaire**, sans avoir à le redéfinir à chaque usage.

De plus, les styles, qui peuvent être définis à l'intérieur d'une page HTML, peuvent (doivent !) être définis dans un **fichier externe qui sera associé à toutes les pages HTML qui en ont besoin**. Ceci permettra de regrouper dans un seul fichier toute la « charte graphique » et les aspects de présentation d'un site Web, ce qui bien sûr simplifiera sa maintenance et son évolution.

Les feuilles de style CSS sont constituées de formats que l'on définit pour certains éléments HTML ou pour un ensemble d'éléments HTML. Pour choisir cet ensemble d'éléments HTML auquel doit s'appliquer un format, on utilise dans le code CSS ce qu'on appelle des **sélecteurs**. Pour chaque sélecteur (qui permet de sélectionner les éléments HTML concernées par le style), on va définir un style de présentation en utilisant un certain nombre de **propriétés/attributs** auxquelles on va affecter des **valeurs**. Par exemple, on peut définir un format pour les balises **<h1>** pour lesquelles on va fixer les propriétés telles la taille de police, la couleur de police, etc...

Les feuilles de style offrent de multiples possibilités. Ce qui est décrit ici concerne uniquement les utilisations les plus courantes. Pour plus de détails, reportez vous à l'aide en ligne de Microsoft, par exemple, ou aux tutoriaux et traductions des standards CSS (documents en ligne). Retenez que maîtriser CSS relève des métiers de « Web Designer » et « Intégrateur Web », et non des développeurs informatique !

## II UTILISER LES FEUILLES DE STYLES EN HTML

Plusieurs méthodes sont utilisables pour intégrer des feuilles de style en HTML :

### Définir globalement des formats dans un fichier HTML

Les balises HTML `<style...> ... </style>` permettent de contenir la définition des formats CSS. Cette définition se positionne dans l'en-tête (**head**) du document. Pour les navigateurs (antiques) qui n'interprètent pas l'élément `style`, vous pouvez insérer le contenu dans un commentaire HTML (`<!-- ... -->`)

Exemple:

```
<html>
<head>
<title>Titre du fichier</title>
<style type="text/css">
<!--
/* ... ici sont définis les formats ... */
-->
</style>
</head>
<body>....

</body>
</html>
```

### Définir un format global dans un fichier CSS séparé

On peut définir les formats dans un (des) fichier(s) texte (**.css**) séparé(s) qu'il suffira de relier aux fichiers HTML. Toute mise à jour du fichier .css ira se répercuter sur toutes les pages HTML utilisant cette feuille de style. C'est la meilleure solution lorsque l'on développe une application Web professionnelle car on réalise une séparation entre les données et leur présentation.

L'appel du fichier .css s'effectue à l'aide de la balise HTML `<link>` :

**`<link rel="stylesheet" type="text/css" href="styles.css" />`**

Exemple:

```
<html>
<head>
<title>Titre du fichier</title>
<link rel="stylesheet" type="text/css" href="formats.css" />
</head>
<body>
</body>
</html>
```

NB : depuis CSS3, l'attribut *media* de la balise `<link />` permet de spécifier quelle feuille de style utiliser en fonction du type de périphérique, ce qui facilite l'auto-adaptation de la page au navigateur (notion de *responsive design*).

Il est possible de combiner l'appel à un (des) fichier(s) .css externe(s) et la définition d'un style local **<style>**. Dans ce cas, en cas de conflit, les formats directement définis dans **< style >** sont prioritaires.

### Définir des formats localement dans un élément HTML

L'application d'un style peut se faire directement dans une balise HTML et ne sera applicable qu'à ce seul élément (et éventuellement aux éléments HTML contenus dans l'élément stylé - notion d'héritage des styles en cascade-).

Exemple:

```
<html>
<head>
<title>Titre du fichier</title>
</head>
<body>
<h1 style="color:blue">...</h1>
</body>
</html>
```

## III DEFINITION DES FORMATS (SELECTEURS)

Considérons la déclaration de style suivante :

```
<style type="text/css">
<!--
body { background-color:#FFFFCC;
margin-left:100px; }
h1 { font-size:48pt;
color:#FF0000;
font-style:italic;
border-bottom:solid thin black; }
p,li { font-size:12pt;
line-height:14pt;
font-family: Arial;
letter-spacing:0.2mm;
word-spacing:0.8mm;
color:blue; }
-->
</style>
```

En syntaxe CSS, on déclare des **sélecteurs** (ici body, h1, p, li) puis leur format. Par "sélecteur" on entend ce qui se trouve devant les accolades. Le sélecteur désigne ce à quoi les définitions qui suivent s'appliquent. Les sélecteurs peuvent être une (ou plusieurs) balises HTML. C'est le cas de l'exemple ci-dessus.

**elem {...}** : concerne tous les balises **<elem>** .

Il est aussi possible d'avoir des sélecteurs beaucoup plus élaborés comme par exemple :

- **parent elem {...}** : concerne tous les éléments **<elem>** contenus dans un élément **<parent>**  
Ex : `a img {...}` concerne toutes les images sensibles
- **parent > elem {...}** : concerne tous les éléments **<elem>** fils direct d'un élément **<parent>**  
Ex : `a>img {...}` ne concerne pas une image sensible définie par le code HTML `<a href...><p>...<img.../>...</p></a>`
- **elem[attr="val"] {...}** : concerne tous les éléments **<elem>** dont la valeur de l'attribut **attr** est **val**.  
Ex : `img[align="right"]` ne concerne que les images alignées à droite
- etc..... **voir : [http://www.w3.org/TR/CSS/ \\$4.2.Selector Index](http://www.w3.org/TR/CSS/ $4.2.Selector Index)**

On peut aussi définir des sélecteurs qui nous sont propres (et que l'on reliera à l'élément HTML par son attribut **class**). Ces sélecteurs ne concernent pas forcément une ou un ensemble d'éléments HTML, mais uniquement les éléments qui seront 'marqués' comme devant appliquer le style défini. Le marquage se fait par l'attribut HTML **class**. Par exemple :

```
<html>
<head>
<style type="text/css">
<!--
    .rouge {
        color: red;
    }
//-->
</style>
</head>
<body>
<h3 class="rouge">Titre rouge</h3>
<h3>Titre normal </h3>
<p class="rouge">Texte écrit en rouge</p>
</body>
</html>
```

Dans cet exemple, on définit un format nommé **rouge**, qui va s'appliquer à toutes les balises reliées à ce style. Toutes les balises dans lesquelles on retrouve l'attribut **class="rouge"** appliqueront le style défini. On peut aller encore plus loin en déclinant des styles pour certains éléments HTML : par exemple si on avait défini le sélecteur **p.rouge**, le style ne serait applicable qu'à un élément **<p >...</p>** ayant un attribut **class="rouge"**.

De plus, on peut associer automatiquement un élément HTML à un style nommé **#xxx** si cet élément HTML possède un attribut HTML **id="xxx"** (NB : l'attribut HTML **id** est fait avant tout pour identifier un élément HTML dans une page afin de la manipuler avec JavaScript et il doit être unique dans la page).

Tout cela se résume en quelques règles simples :

- le rendu standard des éléments HTML est toujours pris en compte (*voir iecss.com*) ; il peut être complété/rectifié par des styles CSS ;
- les styles nommés des noms d'éléments HTML s'appliquent automatiquement aux éléments HTML correspondants (`h1{...}`) ;
- les styles nommés comme des déclinaisons d'éléments HTML s'appliquent aux éléments HTML dès que l'attribut `class` est renseigné (`h1.rouge{...}` → `<h1 class="rouge" >`) ;
- les styles nommés librement s'appliquent aux éléments HTML dès que l'attribut `class` est renseigné (`.gras{...}` → `<p class="gras" >`) ;
- les styles nommés librement en `#xxx` s'appliquent aux éléments HTML dès que l'attribut `id` est renseigné (`#footer{...}` → `<div id="footer" >`) ;

## IV LES BALISES <DIV> ET <SPAN>... ET LES AUTRES...

Les deux balises **<div>** et **<span>** permettent d'identifier des parties de document.

**<div>** est utilisé pour des blocs de texte entiers (et génère un saut de ligne), et

**<span>** est utilisé pour des portions de texte (mots ou lettres).

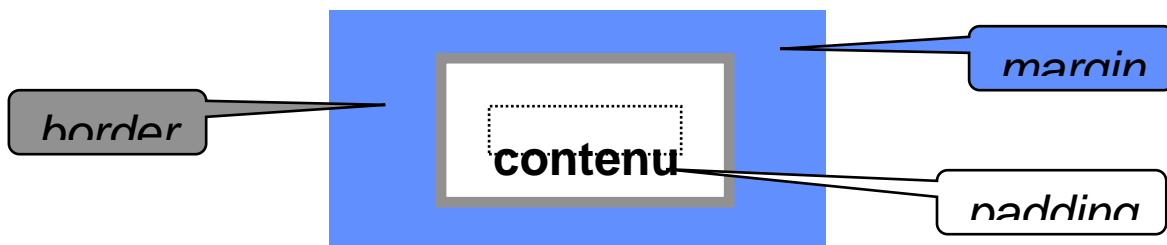
Ces éléments n'ont pas de rendu prédéfini en HTML. Ils peuvent avoir leur propre style (différent des styles des éléments qui les englobent), en utilisant l'attribut **class**. Ainsi, on peut par exemple varier la mise en forme d'un mot particulier (en l'encadrant dans une balise `<span ... class=...>`) placé dans un paragraphe déjà associé à un style (`<p class=...>`).

En XHTML, l'emploi des balises `<div>` est préconisé à la place des `<table>` pour réaliser des mises en page soignées.

En HTML5, les nouvelles balises structurantes `<header>`, `<footer>`, `<nav>`, `<article>`... sont préconisées (éviter la « divite aigüe ») ; ils ne disposent pas de rendu prédéfini et leur présentation doit être entièrement définie en styles CSS.

## V LE MODELE DE « BOITES »

Tout élément HTML peut être vu comme une boîte pouvant contenir d'autres éléments HTML ou du texte. Chaque boîte possède généralement un **contenu** dont CSS peut affecter la **couleur de contenu** (`color`), la **couleur de fond** (`background-color`). Elle dispose de marges externes appelées « **margin** » (entre elle et sa boîte conteneur) et de marges internes appelées « **padding** » (entre ses limites et son contenu). Entre les deux, toute boîte peut recevoir un encadrement (« **border** »). CSS propose une multitude d'attributs pour définir les différentes parties des marges internes et externes ainsi que les caractéristiques des différentes parties de la bordure.



De plus, si cet élément est un '*block*' (voir ci-après), il peut posséder une **largeur** (`width`) et une **hauteur** (`height`).

Entre 2 éléments '*block*', les marges verticales fusionnent au profit de la plus grande. ce principe s'applique aussi bien pour des éléments successifs que imbriqués.

## VI DEFINITION DES PROPRIETES

Pour chaque sélecteur défini, on va attribuer des valeurs à certaines propriétés CSS pour définir la mise en forme. La norme CSS est extrêmement complète et devient complexe à mettre en œuvre en raison du très grand nombre de propriétés/attributs disponibles ; il ne s'agit pas ici d'apprendre tout CSS mais plutôt de repérer les fonctions essentielles de manière à pouvoir mettre en œuvre les principes, pouvoir approfondir le sujet en cas de besoin et utiliser correctement les feuilles de styles fournies.

NB : La création des feuilles de styles ne fait pas partie du cœur de notre métier. Dans le meilleur des cas, le développeur reçoit une charte graphique et une (des) feuille(s) de styles définies par le 'web designer' ou 'l'intégrateur web' et il lui reste à appliquer les styles définis aux éléments HTML des pages qu'il développe.



Voici une liste des principales propriétés et les valeurs les plus usitées :

Type	Propriété	Utilité
<i>Couleurs</i>	<b>color</b>	couleur d'avant-plan
	<b>background-color</b>	La couleur d'arrière plan
	<b>background-image</b>	Une image à utiliser comme fond
<i>Polices</i>	<b>font-family</b>	Une liste de familles de la police à utiliser
	p { font-family: Baskerville, "Heisi Mincho W3", Symbol }	
	<b>font-style</b>	<b>normal, italic, oblique</b>
	<b>font-weight</b>	<b>normal, bold, bolder, lighter ...</b>
	<b>font-size</b>	La taille de la police exprimée soit par une taille absolue déterminée par le navigateur ( <b>xx-small, x-small, small, medium, large, x-large, xx-large</b> ), soit par une mesure relative ( <b>larger, smaller</b> ) soit par une mesure de longueur soit par un pourcentage par rapport à l'élément englobant
	<b>font</b> Ex : p { font: bold italic large Palatino, serif }	Un raccourci pour toutes les propriétés ci-dessus
<i>Texte</i>	<b>text-indent</b>	Indentation de la première ligne d'un bloc
	<b>text-align</b>	Alignement horizontal du texte d'un bloc: <b>left, right, center, justify</b> .
	<b>text-decoration</b>	Attributs additionnels de décoration : <b>underline, overline, line-through, blink</b>
	<b>text-transform</b>	Modification de la capitalisation d'un bloc de texte : <b>capitalize, uppercase, lowercase</b>
<i>Espacement externe et internes</i>	<b>margin-top margin-right margin-bottom margin-left margin</b>	Largeur de marge externe vis-à-vis de l'élément contenant. La valeur est soit une longueur, soit un pourcentage de la boîte englobante. <b>margin</b> est un raccourci pour ces quatre spécifications individuelles.

	<b>padding-top padding-right padding-bottom padding-left padding</b>	Largeur de marge interne de l'élément <b>padding</b> est un raccourci pour ces quatre spécifications individuelles.
<i>Bordures</i>	<b>border-top-width border-right-width border-bottom-width border-left-width border-width</b>	Largeur de bordure. La valeur est soit une longueur soit <b>thin</b> , <b>medium</b> ou <b>thick</b> . <b>border-width</b> est un raccourci pour ces quatre spécifications individuelles.
	<b>border-style</b>	Le style de cadre : <b>dotted</b> (pointillé), <b>dashed</b> (pointillé long), <b>solid</b> , <b>double</b> , <b>groove</b> (creusé), <b>ridge</b> (relief)
	<b>border-top-color border-right-color border-bottom-color border-left-color border-color</b>	Couleur de bordure. <b>border-color</b> est un raccourci pour ces quatre spécifications individuelles.
	<b>border</b> Ex : border : 5px solid black ;	Raccourci pour les trois propriétés précédentes
<i>Listes</i>	<b>list-style-type</b>	Type de numérotation de la liste : <b>disc</b> , <b>circle</b> , <b>square</b> , <b>decimal</b> , <b>lower-roman</b> , <b>upper-roman</b> , <b>lower-latin</b> , <b>upper-latin</b> , ...
	<b>list-style-image</b>	Une image à utiliser comme marqueur d'élément de liste (surcharge list-style-type).
	Ex : ul { list-style-image: url(http://png.com/ellipse.png) }	
<i>Tableaux</i>	<b>caption-side</b>	Position de la légende du tableau : <b>top</b> , <b>bottom</b> , <b>left</b> ou <b>right</b> .
	<b>vertical-align</b>	La hauteur d'une rangée dépend de la taille et du placement des cellules qu'elle contient. Une cellule est donc souvent moins haute que la hauteur de la rangée à laquelle elle appartient. Cette propriété détermine l'alignement vertical d'une cellule par rapport à sa rangée : <b>baseline</b> (aligne la ligne de base texte de la cellule avec la ligne de base de la rangée),

		<b>bottom</b> (le bas de la cellule coïncide avec le bas de la rangée), <b>top</b> , <b>middle</b>
	<b>border-spacing</b>	Espacement des bordures de cellules. La spécification comprend une ou deux longueurs (une seule longueur s'applique à l'espacement horizontal et vertical), deux longueurs correspondent respectivement à l'espacement horizontal et vertical.
	table { border-spacing: 15pt }	
	<b>empty-cells</b>	Attributs de bordure pour les cellules vides : <b>border</b> ou <b>no border</b> .

## VII LE POSITIONNEMENT DES ELEMENTS

Il existe différentes formes de positionnement des éléments qui seront présentées dans les chapitres suivants :

- Positionnement dans le flux (normal)
- Positionnement relatif
- Positionnement flottant
- Positionnement absolu

Ces différents positionnements des éléments vous donnent la possibilité de mieux gérer l'apparence de vos pages Web en fonction de l'écran (notion de *responsive design*).

Propriété	Utilité	Valeur(s)	Exemple
<b>position</b>	Mode de positionnement	<b>absolute</b> = Positionnement absolu, mesuré à partir du bord de l'élément parent; peut défiler. <b>fixed</b> = Positionnement absolu, mesuré à partir du bord de l'élément parent; reste fixe lors du défilement. <b>relative</b> = Positionnement relatif mesuré à partir de	{position:absolute}

		la position de départ de l'élément proprement dit. <b>static</b> = Pas de positionnement spécial, flux normal de l'élément (réglage par défaut).	
<b>top</b>	Position à partir du haut	Valeur en cm, pixel, %...	{ top:3.6cm }
<b>left</b>	Position à partir de la gauche	Valeur en cm, pixel, %...	{left:350px}
<b>bottom</b>	Position à partir du bas	Valeur en cm, pixel, %...	
<b>right</b>	Position à partir de la droite	Valeur en cm, pixel, %...	
<b>width</b>	Largeur	Valeur en cm, pixel, %...	{width:200px}
<b>min-width</b>	Largeur minimale	Valeur en cm, pixel, %...	
<b>max-width</b>	Largeur maximale	Valeur en cm, pixel, %...	
<b>height</b>	Hauteur	Valeur en cm, pixel, %...	{height:100px}
<b>min-height</b>	Hauteur minimale	Valeur en cm, pixel, %...	
<b>max-height</b>	Hauteur maximale	Valeur en cm, pixel, %...	
<b>overflow</b>	Affichage d'élément au contenu trop important pour la taille de la boîte	<b>visible</b> = L'élément sera agrandi de manière à ce que son contenu soit complètement visible dans tous les cas. <b>hidden</b> = L'élément sera coupé s'il dépasse les limites. <b>scroll</b> = L'élément sera coupé s'il dépasse les limites. Le navigateur propose des barres de défilement. <b>auto</b> = Le navigateur Web doit décider d'afficher les barres de défilement en cas de dépassement.	{overflow:auto }
<b>direction</b>	Sens de lecture des éléments	<b>ltr</b> = de gauche à droite. <b>rtl</b> = de droite à gauche.	
<b>float</b>	spécifie qu'un élément doit être retiré du flux normal et placé à la droite ou à la gauche de son bloc conteneur	<b>Left, right, none, inherit</b>	{ float :left }

<b>clear</b>	définit si un élément reste à côté des éléments flottants qui le précèdent ou s'il doit être placé en dessous d'eux (être libéré)	<b>Left, right, both, none</b>	{clear:both}
<b>z-index</b>	Position de la couche en cas de superposition	Valeur numérique. Les éléments dont le numéro est le plus élevé couvrent les éléments avec le numéro le moins élevé.	{ z-index:4 }
<b>display</b>	Mode d'affichage ou plutôt non affichage sans prendre de place	<b>block</b> = impose un bloc - l'élément crée une nouvelle ligne. <b>inline</b> = impose l'affichage en texte - l'élément est affiché dans le cours du texte. <b>inline-block</b> = idem mais dans la limite du conteneur (avec, retours à la ligne automatique) <b>list-item</b> = comme block, mais précédé d'une puce comme dans une liste énumérative. <b>none</b> = L'élément n'est pas affiché et aucune place ne lui sera réservée <b>table, table-cell, table-row...</b> = pour reproduire un tableau HTML	{display :block}
<b>visibility</b>	Affichage ou non affichage avec réservation de place	<b>hidden</b> = Le contenu de l'élément est d'abord caché (non affiché). <b>visible</b> = Le contenu de l'élément est d'abord affiché (réglage normal).	{visibility :hidden}

Par défaut chaque élément a un type d'affichage par rapport au flux HTML :

- Un '**block**' comme <p>, <h2>, <div>... a une largeur par défaut de 100% de la place disponible ; les éléments 'block' se succèdent donc verticalement dans la page Web (du moins, dans leur conteneur).
- Un 'inline-block' comme <input...> possède lui aussi une largeur et une hauteur mais sa largeur par défaut reste ajustée à son contenu de manière à se couler en continu sur la ligne courante.
- Un 'inline' se coule toujours en continu dans le flux HTML et ne possède pas de propriété de largeur ni de hauteur.

La propriété CSS display permet de définir ou redéfinir à l'envie le type d'affichage d'un élément par rapport au flux HTML.

## VII.1 LE POSITIONNEMENT EN FLUX NORMAL

Le positionnement en flux normal correspond à la mise en forme résultant de l'interprétation au fur et à mesure de la transcription des balises de la page HTML. Il résulte de la lecture séquentielle des éléments en partant du début et jusqu'à la fin. Comme dans la lecture d'un texte, il procède 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.

### Boîte de type bloc en flux normal

Par défaut, les boîtes de type bloc seront affichées dans une succession verticale. Dans l'exemple qui suit deux blocs sont représentés par des paragraphes différenciés par la couleur jaune ou rouge.

Le code HTML est extrêmement simple.

```
<p class="jaune">Une boîte jaune</p>
<p class="rouge">Une boîte rouge</p>
```

Le codage de la feuille de style CSS :

```
.jaune {
    background-color: yellow;
}
.rouge {
    background-color: red;
}
```

Le résultat obtenu dans lors de l'affichage dans le navigateur :

Une boîte jaune

Une boîte rouge

Le navigateur traite successivement les deux éléments rencontrés. Comme il s'agit d'éléments de type bloc, il aligne la marge gauche de chaque élément sur la marge gauche de l'élément conteneur, c'est à dire ici le bloc conteneur initial Body. Pour rappel, les principaux éléments créant des boîtes bloc sont :

- l'élément `div` ;
- les titres `h1`, `h2`, `h3`, `h4`, `h5`, `h6` ;
- le paragraphe `p` ;
- Les listes et éléments de liste `ul`, `ol`, `li`, `dl`, `dd` ;
- Le bloc de citation `blockquote` ;
- Le texte pré-formaté `pre` ;
- L'adresse `address`.

### Boîte de type en ligne en flux normal

Remplaçons maintenant les deux paragraphes par des éléments `span` (portion de texte) de type en ligne.

Le code HTML reste extrêmement simple.

```
<p><span class="jaune">Une boîte jaune</span>
<span class="rouge">Une boîte rouge </span></p>
```

CSS3

La feuille de style reste inchangée.

On obtient alors le résultat suivant :

Une boîte jaune Une boîte rouge

Nous nous apercevons alors que les éléments en ligne se succèdent horizontalement.

## VII.2 LE POSITIONNEMENT EN FLUX RELATIF

Le positionnement relatif permet d'inscrire un contenu en flux normal, puis de le décaler horizontalement et/ou verticalement.

Le contenu suivant n'est pas affecté par ce déplacement. Cela peut donc entraîner des chevauchements.

Soit par exemple un positionnement relatif qui produit un décalage vers le haut de 5 pixels et un arrière-plan jaune :

Le code HTML de l'exemple

```
<p>
Début du positionnement en flux normal
<span class="jaune">boîte jaune en position relative</span>
Suite du flux normal</p>
```

Le style CSS :

```
.jaune {
position: relative;
bottom: 5px;
background-color: yellow;
}
```

Résultat

Début du positionnement en flux normal boîte jaune en position relative Suite du flux normal

Pour illustrer le risque de chevauchement, ajoutons un décalage vers la droite en positionnant la boîte à 20 pixels gauche du contenu précédent :

```
.jaune {
position: relative;
bottom: 5px;
left : 20 px ;
background-color: yellow;
}
```

Résultat

Début du positionnement en flux normal boîte jaune en position relative avec décallage gauche Suite du flux normal

Nous constatons alors un chevauchement sur le reste du flux en mode normal qui n'est pas affecté par le positionnement relatif.

### VII.3 LE POSITIONNEMENT FLOTTANT

La position **float** retire une boîte du flux normal pour la placer le plus à droite ou le plus gauche possible dans son conteneur.

Une boîte flottante est donc 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. Ce mécanisme est bien connu en traitement de texte pour écrire du texte autour d'une image. Il faut prendre garde à la position du conteneur et à la position des éléments dans celui-ci.

Un exemple pour illustrer le fonctionnement des boites flottantes

Reprenons l'exemple des boites jaune et rouge précédent. Ajoutons aux styles précédents une règle de positionnement flottant à droite. Ajoutons du texte pour que la présentation soit significative dans l'élément devant être visualisé autour de l'élément flottant:

Pour un meilleur rendu visuel, le texte est justifié et une marge gauche est définie pour la boîte flottante.

```
.jaune
{
background-color: yellow;
text-align: justify;
}
.rouge
{
background-color: red;
float: right;
width: 100px;
margin: 0 0 0 10px;
}
```

Résultat :

Une boîte flottante est donc 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. Ce mécanisme est bien connu en traitement de texte pour écrire du texte autour d'une image. Reprenons l'exemple des boites jaune et rouge. Ajoutons aux styles précédents une règle de positionnement flottant à droite. Copions du texte pour que la présentation soit significative dans l'élément devant être visualisé autour de l'élément flottant.

Il est important de tenir compte du flux HTML et d'ordonner correctement les éléments HTML. L'élément flottant doit précéder l'élément qui l'entoure.

Le positionnement flottant est très utile pour faire du multicolonnage.

L'exemple suivant montre comment diviser l'espace de la page en 3 colonnes de même largeur avec une séparation verticale à gauche.

Il faut toutefois prendre garde au dépassement (overflow) qui peut engendrer un rendu visuel peu satisfaisant (La largeur des 3 colonnes dépasse 100% de la zone d'affichage...)



## La définition des styles :

```
.colonne
{
float: left;
width: 32%;
background-color: yellow;
text-align: justify;
border-left-style : solid;
border-left-color : blue;
border-left-width : 0.2em;
padding-left : 0.3em;
}
.corps
{
background-color: red;
margin: 0;
padding: 0;
}
```

## Le code HTML

```
<body class='corps'>
<div class='colonne'>
<p>Le positionnement en flux normal correspond à la mise en forme résultant
de l'interprétation au fur et à mesure de la transcription des ...</p>
</div>
<div class='colonne'>
<p>Le positionnement relatif permet ...</p>
</div>
<div class='colonne'>
<p>La position float retire ...</p>
</div>
</body>
```

## Le résultat obtenu :

Le positionnement en flux normal correspond à la mise en forme résultant de l'interprétation au fur et à mesure de la transcription des balises de la page HTML. Il résulte de la lecture séquentielle des éléments en partant du début et jusqu'à la fin. Comme dans la lecture d'un texte, il procède 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.	Le positionnement relatif permet d'insérer un contenu en flux normal, puis de le décaler horizontalement et/ou verticalement. Le contenu suivant n'est pas affecté par ce déplacement. Cela peut donc entraîner des chevauchements.	La position float retire une boîte du flux normal pour la placer plus à droite ou le plus gauche possible dans son conteneur. La boîte flottante est donc placée le plus à droite (float: right) ou 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. Ce mécanisme est bien connu en traitement de texte pour écrire du texte autour d'une image. Il faut prendre en compte la position du conteneur et la position des éléments de celui-ci.
--	---	--

## Les problèmes de débordement :

Lorsque nous incluons des éléments flottants dans une subdivision de page <div> sans préciser de taille pour celle-ci (taille des contenus), il se produit des effets non recherchés. Exemple, dans le cas de cette page avec 2 colonnes dans une division dont le fond est vert, le corps de la page ayant un fond rouge, nous obtenons ceci :

Le positionnement en flux normal correspond à la mise en forme résultant de l'interprétation au fur et à mesure de la transcription des balises de la page HTML. Il résulte de la lecture séquentielle des éléments en partant du début et jusqu'à la fin. Comme dans la lecture d'un texte, il procède 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.

Le positionnement relatif permet d'inscrire un contenu en flux normal, puis de le décaler horizontalement et/ou verticalement. Le contenu suivant n'est pas affecté par ce déplacement. Cela peut donc entraîner des chevauchements.

Le style de la division conteneur ne précise pas la taille de celle-ci.

```
.global
{
  background-color: green;
  margin: 0;
  padding: 0;
}
```

Hors nous souhaitons obtenir un fond vert sur toute la division.

Nous allons avoir recours à la propriété **clear** qui interdit le voisinage d'un élément flottant, forçant ainsi l'élément portant cette propriété à se positionner dans le respect de la propriété clear.

Clear admet 3 valeurs : right pas d'élément flottant à droite, left aucun flottant à gauche, both aucun flottant ni à droite ni à gauche.

Dans notre cas, nous allons forcer le position d'un élément quelconque non affiché après les éléments flottants en utilisant la propriété clear :both. L'élément va donc être repositionné dans le flux courant.

Ajout d'une division invisible juste après la deuxième colonne avec le style suivant :

```
#separation
{
  clear: both;
  visibility: hidden;
}
```

Donne le résultat suivant : Le fond vert est bien affiché sous les éléments flottants.

Le positionnement en flux normal correspond à la mise en forme résultant de l'interprétation au fur et à mesure de la transcription des balises de la page HTML. Il résulte de la lecture séquentielle des éléments en partant du début et jusqu'à la fin. Comme dans la lecture d'un texte, il procède 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.

Le positionnement relatif permet d'inscrire un contenu en flux normal, puis de le décaler horizontalement et/ou verticalement. Le contenu suivant n'est pas affecté par ce déplacement. Cela peut donc entraîner des chevauchements.

## VII.4 LE POSITIONNEMENT ABSOLU OU FIXE

Une boîte en positionnement absolu peut être placée n'importe-où dans le code HTML et s'afficher à l'endroit de votre choix. Ceci s'avère très utile en particulier pour :

- Placer les menus de navigation en fin de page, pour améliorer l'accessibilité de votre site en donnant un accès immédiat à son contenu dans les navigateurs textes, tout en les faisant apparaître en haut de page ou encore dans une colonne pour les navigateurs graphiques ;
- Créer plusieurs colonnes ou plusieurs boîtes au positionnement indépendant de l'ordre dans lequel elles se trouvent en HTML.

Le fonctionnement de la position absolue

Le positionnement absolu « retire » totalement du flux le contenu concerné : sa position est déterminée par référence aux limites du conteneur. Celui-ci peut-être :

- une boîte elle-même positionnée (position relative ou absolue) ;
- le bloc conteneur initial, à défaut de boîte positionnée, c'est à dire en pratique le plus souvent la fenêtre du navigateur (le parent le plus ancien ou racine)

Dans l'exemple qui suit nous avons 3 boîtes qui sont positionnées de manière absolue par rapport à une division (le trait bleu) positionné de manière relative. Le point de référence sera donc la division. Si la division n'avait pas été positionnée, le point de référence aurait été le cadre de la fenêtre de navigation.

Les principaux styles définis :

Le point de référence

```
.repere
{
  background-color: blue;
  position: relative;
  top: 50px;
  left : 20 px;
  height: 0.5em;
}
```

```
#colonne1
{
position : absolute;
top: 2em;
Left: 2em;
...
}
```

```
#colonne2
{
position : absolute;
top: 4em;
Left: 4em;
...
}
```

```
#colonne3
{
position : absolute;
top: 6em;
Left: 6em;
...;
}
```

Le positionnement en flux normal correspond à la mise en forme

résult

des t

des é

lectu

» de

gauche a

en largeur

Le positionnement relatif permet d'inscrire un contenu en flux

La position float retire une boîte du flux normal pour la placer le plus à droite ou le plus gauche possible dans son conteneur. Une boîte flottante est donc 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. Ce mécanisme est bien connu en traitement de texte pour écrire du texte autour d'une image. Il faut prendre garde à la position du conteneur et à la position des éléments dans celui-ci.

## Profondeur des éléments

Nous pouvons indiquer un ordre de profondeur différent de celui retenu (ordre dans le flux). La propriété **z-index** s'apparente au travail avec des couches ou calques.

Ainsi, nous allons, pour l'exercice, faire apparaître au premier plan la colonne1 en jaune ; en deuxième plan la rouge et en dernier la zone violette.

Nous obtenons pour résultat :

Le positionnement en flux normal correspond à la mise en forme résultant de l'interprétation au fur et à mesure de la transcription des balises de la page HTML. Il résulte de la lecture séquentielle des éléments en partant du début et jusqu'à la fin. Comme dans la lecture d'un texte, il procède 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.

Le positionnement relatif permet d'inscrire un contenu en flux. La position float retire une boîte du flux normal pour la placer le plus à droite ou le plus gauche possible dans son conteneur. Une boîte flottante est donc 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. Ce mécanisme est bien connu en traitement de texte pour écrire du texte autour d'une image. Il faut prendre garde à la position du conteneur et à la position des éléments dans celui-ci.

Les valeurs de z-index n'ont aucune signification en absolu. Seule compte leur position relative, l'élément le plus élevé étant affiché au dessus des autres.

La position fixe

Il s'agit uniquement d'une spécialisation du positionnement absolu, le contenu concerné est retiré totalement du flux. Mais il est cette fois positionné uniquement par rapport aux limites de la zone de visualisation, autrement dit la fenêtre du navigateur. Le défilement de la page n'a aucun effet sur un contenu en position fixe. Le fait de mettre un z-index supérieur lui permettra d'être toujours visible.

La position fixe doit être utilisée avec précaution, car cela fige une partie de la zone d'affichage.

## VIII LES UNITES DE MESURE

CSS offre de nombreuses unités de mesure pour définir les dimensions (d'élément, de marge, de texte...), au contraire de HTML très limité en la matière :

- Unités métrique absolues comme **cm**, **mm** ou **in**(che)
- **pt** pour point typographique (=1/72 pouce soit 1/3 mm)
- **px** pour pixel
- **%** pour une proportion de la place disponible
- **em**, mesure relative à la taille de la police de l'élément parent

La maîtrise de la présentation par CSS repose sur celle de ces unités de mesure, en particulier les unités relatives *em* et *%*.

## IX LES PSEUDO-ATTRIBUTS

Les pseudo-attributs s'appliquaient principalement à l'élément HTML `<a...>` ; ils permettaient de préciser les couleurs sur les liens. Les pseudos-attributs sont les suivants :

- **a:link** = pour les liens aux pages qui n'ont pas encore été visitées
- **a:visited** = pour les liens aux pages qui ont déjà été visitées
- **a:hover** = pour les liens sur lesquels l'utilisateur passe avec la souris
- **a:active** = pour les liens en train d'être cliqués
- **a:focus** = pour les liens qui deviennent actifs (focus)

Exemple :

```
<style type="text/css">
  a:hover { font-weight:bold; color:#E00000; text-decoration:none }
</style>
```

Tout élément HTML peut maintenant « réagir » au survol de la souris grâce au pseudo-élément :hover, même s'il n'est pas une zone cliquable. Le développeur peut de cette manière ajouter un peu d'animation aux pages à peu de frais.

On peut encore modifier la forme du curseur, par l'intermédiaire de l'attribut **cursor**. Les principales valeurs possibles sont : **auto**, **crosshair**, **hand**, **move**, **resize**, **text**, **wait**, **help**..... Utilisé en combinaison avec le pseudo-attribut :hover, cela permet de modifier complètement l'apparence par défaut des liens hypertexte.

## X LES OMBRAGES

CSS3 apporte une standardisation des ombrages, applicables aussi bien pour le texte que pour toute boîte conteneur.

Propriété	Utilité	Valeur(s)	Exemple
<b>box-shadow</b>	Ombre portée par la boîte conteneur	4 paramètres séparés par un espace : décalage à droite, décalage en bas, taille de l'ombre, couleur de l'ombre	{box-shadow: 5px 5px 10px lightblue;}
<b>text-shadow</b>	Ombre portée par le texte	idem	

CSS3 est défini par un ensemble complexe de standards (des *chapitres*) qui sont spécifiés progressivement par les équipes du W3C ; ainsi certaines fonctionnalités sont déjà bien stabilisées alors que d'autres ne sont qu'à peine ébauchées. Et on parle déjà des 'sélecteurs CSS4'...

(📖 <http://inserthtml.developpez.com/tutoriels/css/specifications-css4/>)

Pendant les nécessaires périodes de transition, chaque moteur de navigateur définit et implémente sa propre vision de la fonctionnalité avant que le standard ne soit stabilisé ; ainsi pour définir un ombrage, la propriété CSS standard est `box-shadow`, mais le monde FireFox a de plus inventé `-moz- box-shadow`, celui de Chrome, `-webkit- box-shadow` et opera `-o- box-shadow` quand Internet Explorer proposait `-ms- box-shadow` ... Au final, pour raison de compatibilité avec d'anciens navigateurs, on écrit bien souvent 5 règles de styles identiques pour s'assurer que chacun y trouvera une règle applicable ! Ceci est valable pour toutes les propriétés CSS3 qui suivent.

Exemple :

`box-shadow: 5px 5px 10px lightblue`

**Boîte ombrée**

## XI LES ANGLES ARRONDIS

CSS3 apporte une standardisation des angles arrondis applicables pour toute boîte conteneur dotée d'une bordure.

On peut définir une valeur générale ou éventuellement pour chacun des coins de la boîte ; les arrondis peuvent être circulaires ou elliptiques.

Propriété	Utilité	Valeur(s)	Exemple
<b>border-radius</b> <b>border-top-left-radius</b> <b>border-bottom-right-radius</b> ...	Arrondis circulaires des angles de la bordure	Taille en pixels, cm, %...	{border-radius: 10 px;}
	Arrondis elliptiques des angles de la bordure	2 paramètres séparés par un espace : largeur et hauteur de l'arc d'ellipse	{border-top-left-radius: 20px 10px ; }

NB : dans l'attente du support complet de CSS3 par tous les navigateurs, il est préférable de compléter cet attribut standard par les attributs similaires spécifiques à chaque navigateur.

### Exemples :

`border-radius: 10 px;`

**Titre arrondi circulaire**

`border-top-left-radius: 20px 10px ;`  
`border-bottom-right-radius: 20px 10px`

**Titre arrondi elliptique**

## XII LES FONDS DEGRADÉS


CSS3 apporte une standardisation des fonds dégradés applicables pour toute boîte conteneur. Le dégradé peut être aussi bien linéaire d'un bord à l'autre, en horizontal, que radial, depuis le centre vers les bordures.

Propriété	Utilité	Valeur(s)	Exemple
<b>background : linear-gradient (...)</b>	Fond linéaire de gauche à droite ou de droite à gauche	3 paramètres séparés par virgule : direction, couleur-début, couleur-fin	{background : linear-gradient(to right, LightBlue, blue ); }
<b>background : radial-gradient (...)</b>	Fond dégradé du centre vers les bordures	3 paramètres séparés par virgule : position-centre, couleur-début, couleur-fin	{background : radial-gradient(ellipse at center, LightBlue, blue ); }

NB : dans l'attente du support complet de CSS3 par tous les navigateurs, il est préférable de compléter cet attribut standard par les attributs similaires spécifiques à chaque navigateur.


#### Exemples :

```
background : linear-gradient(to right, LightBlue, blue );
```



Dégradé linéaire

```
background : radial-gradient(ellipse at center, LightBlue, blue );
```



Dégradé elliptique

## XIII LA TRANSPARENCE

CSS3 apporte une standardisation dans la définition de la transparence d'un élément HTML, applicable à tout élément (texte, image, boîte...). L'usage le plus fréquent est le placement d'un logo en fond (de page, de tableau, de boîte...) qui sera partiellement masqué par le contenu (de la page, du tableau, de la boîte...).

Attribut CSS : **opacity** : ayant pour valeur un nombre compris entre 0.0 (=totalement transparent) et 1.0 (=opaque).

Ex :

```
body { /* image en fond de page */  
    background-image:url("logoAfpa2010.png");  
    background-attachment:fixed;  
    background-position:center center;  
    background-repeat:no-repeat;  
}  
h2 { /* titre avec légère transparence */  
    background : linear-gradient(to right, #FFBBFF, magenta );  
    border: ridge purple;  
    padding: 5px;  
    opacity: 0.8;}
```





## XIV LES TRANSFORMATIONS, TRANSITIONS ET ANIMATIONS

CSS3 apporte une standardisation des transformations d'objets (déformations, rotation et translation), de l'animation de certains attributs et même de la définition de certaines animations simples.

Le but recherché est encore et toujours de définir des opérations de haut niveau avec un langage simple, sans utiliser d'images ou d'animations lourdes à transférer sur les réseaux.

Il faut bien reconnaître que le support de ces nouvelles fonctionnalités par les navigateurs est encore très pauvre et balbutiant ; il reste donc de beaux jours à vivre aux techniques Gif animé et Flash...



Pour aller plus loin :

<http://debray-jerome.developpez.com/articles/les-transformations-en-css3/>

Et le magnifique générateur de syntaxe CSS3 : <http://www.css3maker.com/>

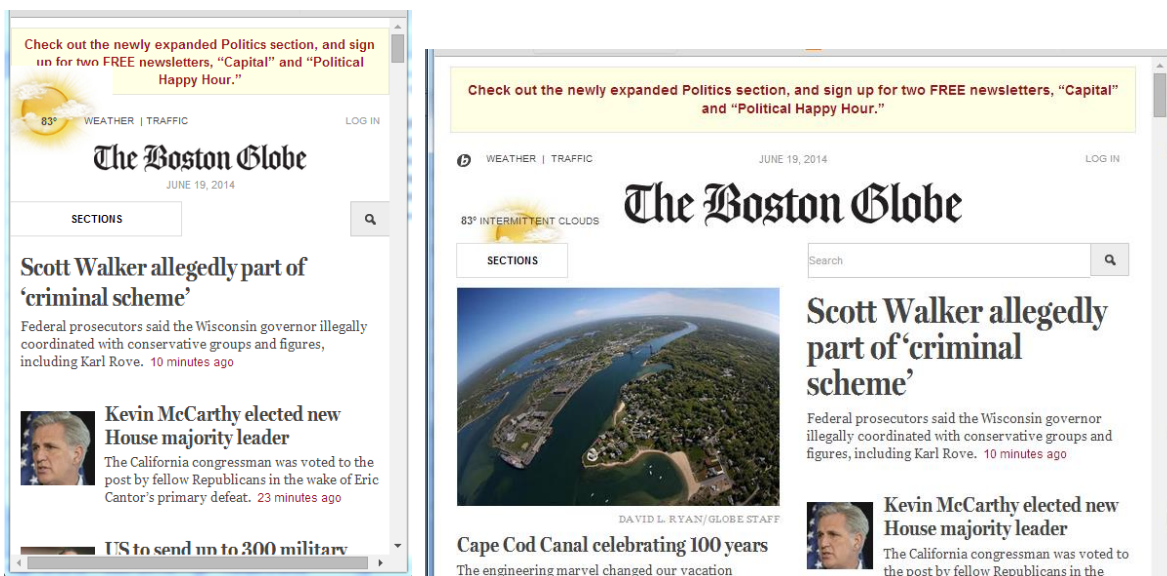
## XVLES JEUX DE STYLES DIFFERENTS SELON LE PERIPHERIQUE

Depuis toujours, CSS permet de définir différents jeux de styles à utiliser selon le périphérique de destination, écran ou imprimante à l'origine.

Avec la multiplication des types de périphériques pouvant afficher une page Web (écrans standards ou larges, tablettes, smartphones, mais aussi terminal braille voire lunettes 3D !), il était nécessaire que la page Web puisse s'adapter au périphérique utilisé. CSS3 introduit la notion de « media query » qui permet de regrouper des jeux de styles à utiliser selon le cas.

Par exemple dans le jeu de styles définis à l'intérieur de **@media screen {...}**, tous les éléments seront affichable mais dans le jeu de styles **@media print{...}** les boutons ou liens de navigation ne seront pas reproduits (display :none ;).

Mais les media queries vont bien au-delà et la mise en page peut être totalement réajustée en fonction du périphérique ; c'est ce qu'on appelle le *responsive design* :



## **Etablissement référent**

*Afpa Pont de Claix*

## **Equipe de conception**

*J.C. Rigal, Jacques Gereys et R.Lecu  
(actualisation B.Hézar – Afpa Nice)*

## **Reproduction interdite**

Article L 122-4 du code de la propriété intellectuelle.  
«toute représentation ou reproduction intégrale ou partielle faite sans le  
consentement de l'auteur ou de ses ayants droits ou ayants cause est  
illicite. Il en est de même pour la traduction, l'adaptation ou la reproduction  
par un art ou un procédé quelconques.»

Date de mise à jour juin 2013  
afpa © Date de dépôt légal juin 2013

