



**CSS 3**

# Plan de cours CSS3

- 1.Les bases**
- 2.Les premiers pas**
- 3.Les propriétés des CCS2**
- 4.Les textes**
- 5.Les arrières plans**
- 6.Les dimensionnements**
- 7.Les bordures**
- 8.Les marges**
- 9.Les boîtes**
- 10.Les positionnements**
- 11.Les nouveautés du CSS3**



# 1 - Les bases

# **1- Sommaire**

- Position du problème
- Qu'est-ce que les CSS ?
- Un exemple d'utilisation Historique
- Spécificités du CSS2

# 1.1- Position du problème

**Le HTML a des limitations importantes :**

- Mise en forme de la présentation :
  - Mise en page difficile
  - Limitation des possibilités
- Maintenance d'un site :
  - Fastidieuse
  - Prend beaucoup de temps
  - A l'origine de l'arrêt de beaucoup de sites !
- Nécessaire séparation de la présentation et des données.

# 1.2- Qu'est-ce que les CSS ?

**CSS ?**

- "Cascading Style Sheet",
- Feuilles de style en cascade

**Objet : rendre cohérents et homogènes les sites Web et faciliter leur maintenance.**

**Toute modification d'une feuille de style est répercutée sur l'ensemble des pages où elle s'applique.**

**Utilisation pour la mise en forme visuelle et sonore :**

- Des polices de caractères
- Des marges et des bordures
- Des couleurs
- Du positionnement des différents éléments dans la page.

# 1.3- Un exemple d'utilisation

Texte HTML sans feuille de style, pour chaque balise rencontrée :

```
<h1 align="center">  
<font color="#0000ff" face="arial">  
    <b>Ceci est un titre bleu, en police arial, gras, centré</b>.  
</font>  
</h1>
```

Style équivalent (une seule occurrence pour toutes les pages)

```
h1.style1 {  
    text-align: center;  
    color: #0000ff;  
    font-family: arial;  
    font-weight: bold; }
```

Contenu du texte HTML utilisant la feuille de style

```
<h1 class="style1">Ceci est un titre bleu, en police arial, gras, centré.</h1>
```

- Comparer (et noter l'attribut class !)

## 1.4- Quelques remarques

**Il y a plus d'une centaine de propriétés !**

- Certaines ne sont valables que pour un nombre limité de balises
- D'autres s'appliquent à l'ensemble des balises.

**Il existe des problèmes de compatibilité de navigateurs**

**Il n'est pas conseillé d'utiliser les versions de navigateurs suivants:**

- Netscape : 4.x et en dessous
- Internet Explorer : 4.x et en dessous
- Problèmes éventuels avec IE 5.5

**Eviter les autres versions de navigateurs, les feuilles de style ne marcheront tout simplement pas.**

# 1.5- Historique

## La première version : CSS1

- Internet Explorer, Netscape en retard
- Recommandation du W3C : 17/12/96, révisée 11/01/1999

## • La deuxième version : CSS2

Recommandation du W3C : 12/05/98

## • La troisième version : CSS3

recommandation du W3C : 21/05/2001

## • Projet de : CSS4

Projet (draft [= *projet*]) : 2009



## 2 - Les premiers pas

## 2- Sommaire

- Les constituants d'une règle
- La déclaration d'un style
- Les différents types de sélecteurs Les sous-classes de format
- Les formats indépendants
- Les pseudo-éléments
- Les boîtes et bordures
- Les unités de mesures en CSS
- Les couleurs en CSS

## 2.1- Les constituants d'une règle

Ceci est une règle :

```
h1.style1 {  
    text-align: center;  
    color: #0000ff;  
    font-family: arial;  
    font-weight: bold; }
```

### Eléments d'une règle

#### Sélecteur

- **h1** = sélecteur d'élément
- **style1** = sous-classe (facultatif)

#### Déclaration :

- **text-align** = propriété
- **center** = valeur

La déclaration de cette règle possède plusieurs propriétés.

## 2.1- Les constituants d'une règle

Supposons qu'une propriété est définie deux fois pour un même élément. Il semblerait que ce soit la dernière la propriété css qui prend le dessus et les autres se cumulent.

```
<style type="text/css">
h1 { font-size:12px; }
.nom { font-size:15px; text-decoration:underline; }
</style>
```

- Si vous ne voulez pas que la propriété de la CLASS prenne le dessus, utilisez !important.

```
<style type="text/css">
h1 { font-size:12px !important; }
.nom { font-size:15px; text-decoration:underline; }
</style>
```

Attention ! : **important** : n'est pas reconnu par les anciens navigateurs et est à utiliser dans les cas exceptionnel.

## 2.2- La déclaration d'un style

- Trois façons de faire :
- Dans le document lui-même
  - Avec l'attribut **style** (toutes balises)
- Avec la balise HTML **<style>** dans le "header"

```
<html>
<head>
<title>Exemple de feuille de
style</title> <style type="text/css">
p {
color: blue;
}
</style>
</head>
```

## 2.2- La déclaration d'un style

### Ø Dans un fichier distinct, souvent à la racine du site

#### § Contenu du fichier "appliWeb.css"

```
/* Ceci est un commentaire facultatif */  
h1.style1 {  
    text-align: center;  
    font-family: arial;  
    color: red;  
}
```

#### § Appel de ce fichier dans une page Web

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

#### § La valeur de href est le chemin d'accès au fichier, comme pour une image par exemple.

## 2.3- Les différents types de sélecteurs

### Ø Sélecteur d'éléments

- Élement {déclaration;}

### Ø Sélecteur d'éléments imbriqués

- Élement\_X élément\_Y {déclaration ;}

Ne s'applique à Y que si Y est imbriqué dans X.

### Ø Sélecteur d'éléments père-fils

- Élement\_X > élément\_Y {déclaration ;}

Ne s'applique que si Y est le fils direct de X.

### Ø Sélecteur d'éléments consécutifs

- Élement\_X + élément\_Y {déclaration ;}

Ne s'applique que si X suit immédiatement Y.

### Ø Sélecteur d'éléments ayant un attribut particulier

- Élement\_X [attr]{déclaration ;}

Ne s'applique que si X possède l'attribut *attr*.

### Ø Sélecteur d'éléments ayant un attribut particulier

- Élement\_X [attr="val"]{déclaration ;}

Ne s'applique que si X possède l'attribut *attr* et sa valeur est *val*.

## 2.4- Les sous-classes de format

- Ø Sert à donner plusieurs formats d'affichage possible à une même balise
  - § Exemple :  
`h1.style1 h1.style2`
- Ø Une sous-classe hérite des caractéristiques de mise en forme de la balise
- Ø La sous-classe permet de rajouter des caractéristiques supplémentaires.
- Ø On peut aussi avoir des sous-classes indépendantes des balises, sans sélecteur :

**.style3**

Cette sous-classe pourra s'appliquer à plusieurs sélecteurs :

`<p class="style3">, <h1 class="style3">, <li class="style3">...`

## 2.5- Les formats indépendants

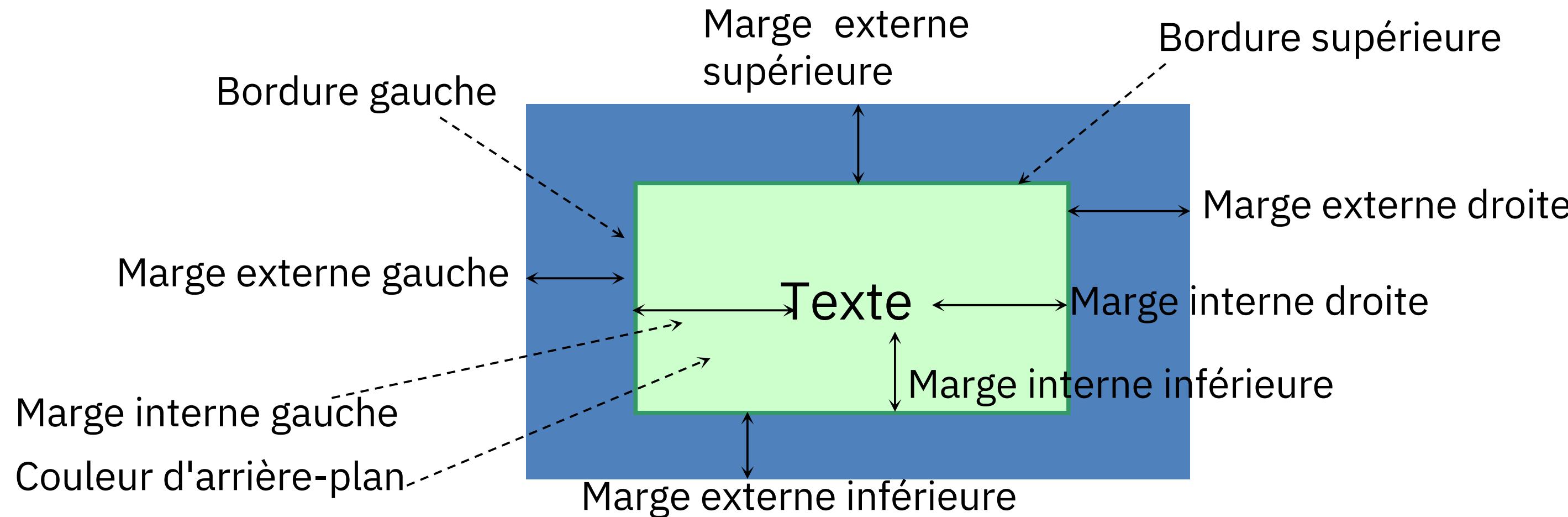
- Ø Permet de définir un style indépendamment de la balise
- Ø HTML Définit dans la feuille de style par un **#** devant le `#bleu {color: blue;}`  
sélecteur :
- Ø Appelé dans la page par l'attribut *id* et non par *class* :  
`<p id="bleu">Ce paragraphe va être écrit en bleu</p>`
- Ø Préférable aux sous-classes indépendantes du paragraphe
- Ø 2.4. Note :
  - § On peut avoir à la fois *class* et *id* dans une même balise !

## 2.6- Les pseudo-éléments

- Ø Permettent d'affecter un style à des éléments qui ne peuvent pas être représentées par un sélecteur habituel.
- Ø En CSS2, Il n'en existe pas encore beaucoup. CSS3 en ajoute plusieurs dizaines (voir chapitre 11.7)
- Ø Usage :
  - § Couleur des liens
  - § Apparence de la première ligne d'un paragraphe
  - § Apparence de la lettrine (première lettre d'un paragraphe)

## 2.7- Les boîtes et bordures (1)

Ø **TOUS** les éléments HTML s'inscrivent dans une boîte dont on peut définir les marges (intérieure, extérieure, gauche, droite, dessus, dessous), la bordure, la couleur de fond...



## 2.7- Les boîtes et bordures (1)

- Ø Vous pouvez en feuille de style css spécifier des marges à un bloc.  
Il existe deux types de marges, les marges intérieures (padding) et les marges extérieures (margin).



## 2.7- Les boîtes et bordures (2)

- Ø La balise **<table>** est souvent utilisée en HTML pour la mise en page.  
A tort !
- Ø Les CSS2 utiliseront plutôt **<div>** sauf bien sûr pour afficher un vrai tableau !
- Ø Il y a une correspondance entre les attributs des balises et les déclarations des CSS2
- Ø On peut placer une boîte n'importe où dans une page.
- Ø Pour les bordures, on peut choisir :
  - § L'épaisseur
  - § La couleur,
  - § Le style,
  - § Le côté concerné

## 2.8- Les unités de mesure

### Ø Données absolues

- § **in** = pouce (2,54 cm)
- § **pt** = point (1/72ème de pouce, environ 0,0352 mm)
  - Attention, point Didot français <> point USA !
- § **pc** = pica (12 points)
- § **mm** = millimètre
- § **cm** = centimètre

### Ø Données relatives

- § **em** = unité relative à la hauteur d'une police donnée
- § **ex** = hauteur entre la ligne de base et le haut du caractère
- § **px** = pixel, fonction de la résolution de l'écran de l'utilisateur
- § **%** = pourcentage par rapport à la norme de l'élément HTML

### Ø Le séparateur décimal est le point, pas la virgule.

## 2.9- Les couleurs en CSS

Ø Définies de la même façon qu'en HTML, avec le système RVB :

- § R= rouge de 0 à FF(FF = valeur hexadécimale de 255)
- § V = vert de 0 à FF
- § B= bleu de 0 à FF
- § Codage en jaune
  - **style="color: #FFFF00" ou #FF00 (chaque digit étant dupliqué)**

Ø Autre méthode en CSS2 : rgb (green, red, blue)

- § Codage pour le jaune
  - **style="color: rgb(255,255,0)"**
  - **style="color: rgb(100%,100%,0)"**

Ø CSS3 : rgba (green, red, blue, alpha)

- § Codage pour le jaune avec 50% d'opacité
  - **style="color: rgba(255,255,0, 0.5)"**

## 2.10- Notion d'héritage

Ø La propriété de l'élément prend la valeur héritée du "parent"

§ Exemple :

```
<html><head><title>Héritage</title>
<style type="text/css">
body { font-family: arial; }
div { font-size: 12px; }
</style>
<head>
<body>
<div>
    <p>Ce paragraphe hérite des propriétés des éléments
body et div : police arial de 12px</p>
</div>
```

Ø Certains éléments ont leur propre style et, par conséquent, n'hérite pas des propriétés de leur parent (par exemple, la balise a).

§ Pour que l'élément `<a>` hérite de la couleur du texte de son parent, on attribut la valeur `inherit` à la propriété `color`.



## 3 - Les propriétés des CSS

### 3- Sommaire

- Ø Les thèmes de propriétés abordées
- Ø Les 16 couleurs de base (rappel)
- Ø Les pseudos-éléments

## 3.1- Les thèmes des propriétés abordées

- Ø **Les couleurs**
- Ø **Les polices de caractères et textes**
- Ø **Les arrières-plans (*background*)**
- Ø **Les dimensionnements**
- Ø **Les bordures**
  - § Couleur
  - § Epaisseur
  - § Style
- Ø **Les marges**
  - § Marges externes (*margin*) Marges internes (*padding*)
- Ø §
- Ø **Les boîtes**
- Ø **Les positionnements**

## 3.2- Les 16 couleurs de base

|            |         |         |
|------------|---------|---------|
| Bleu clair | aqua    | #00FFFF |
| Noir       | black   | #000000 |
| Bleu       | blue    | #0000FF |
| Mauve      | fuchsia | #FF00FF |
| Gris       | gray    | #5D5D5D |
| Vert       | green   | #005D00 |
| Vert clair | lime    | #00FF00 |
| Marron     | maroon  | #5D0000 |

### 3.3- Les 16 couleurs de base

|            |        |         |
|------------|--------|---------|
| Bleu marin | navy   | #00005D |
| Vert olive | olive  | #5D5D00 |
| Pourpre    | purple | #5D005D |
| Rouge      | red    | #FF0000 |
| Argent     | silver | #A8A8A8 |
| Cyan foncé | teal   | #005D5D |
| Blanc      | white  | #FFFFFF |
| Jaune      | yellow | #FFFF00 |

## 3.4 Les pseudo-éléments du CSS2.1

| Pseudo-         | Définition  |
|-----------------|---|
| élément :before | permet d'insérer un contenu avant un élément et de le styler « à la volée »   |
| :after          | permet d'insérer un contenu après un élément et de le styler « à la volée »   |
| :first-letter   | permet de styler la première lettre d'un élément  |
| :first-line     | permet de styler la première ligne d'un bloc  |
| :link           | cible les liens non visités   |
| :hover          | cible un élément pointé visuellement (survolé) grâce à une souris par exemple   |
| :active         | cible un élément activé par l'utilisateur, par exemple au moment du clic sur le bouton gauche de la souris. Intervient entre la pression sur le bouton et son relâchement               |
| :focus          | cible un élément pointé physiquement, grâce à la touche tab du clavier par exemple, ou après le relâchement du bouton gauche de la souris sur un élément pouvant être ciblé de la sorte |
| :visited        | cible un lien déjà visité par l'utilisateur   |
| :lang()         | cible un élément en fonction de sa langue (spécifié par l'attribut lang)  |
| :first-child    | cible le premier élément enfant   |



## 4- Les textes

## 4- Sommaire

### Ø Polices de caractères

- § **font-family**
- § **font-size**
- § **font-style**
- § **font-weight**

### Ø divers

- § **text-align**
- § **text-decoration**
- § **text-transform**
- § **line-height**
- § **vertical-align**

### Ø Utilisation de la balise <span>

# 4.1- font-family

Ø Famille de la police de

Ø caractères Valeurs possibles

*Nom\_de\_police :*

-

§ **serif**

- *arial, times roman, comic sans MS...*

§ **sans-serif**

- Police avec empattement : Times New Roman, Garamond...

§ **monospace**

- Police sans empattement : Arial, helvetica, Futura, Gill Sans

§ **cursive**

- Police non proportionnelle : Courier New...

§ **fantasy**

- Police de type cursive

§ **inherit**

Police de type fantaisie

Ø § **Exemple**

§ **font-family: arial, helvetica, sans-serif;**

## 4.2- font-size

Ø Taille de la police de

Ø caractère Valeurs possibles :

§ Valeurs numériques :

- 10 px, 12pt...

§ Valeurs prédéfinies :

- xx-small, x-small, small, medium, x-large, xx-large

§ Valeurs relatives :

- Dépend de l'élément courant : smaller, larger, 50%...

§ Inherit

Ø Exemples

§ **font-size:**

§ **smaller; font-size: 12px;**

## 4.3- font-style

Ø Style de la police de

Ø caractère Valeurs possibles :

- § normal
- § Italic
  - Caractères en italique
- § Inherit

Ø Exemple :

§ **font-style: italic**

## 4.4- font-weight

Ø

Ø

- § **Mise en gras du texte. Valeurs possibles**
- §      **normal**
- §      **bold**
- § - **Caractère gras**
- §      **bolder**
- § - **La graisse augmente**
- §      **lighter**
- § - **La graisse diminue**
- §      **Valeur\_numérique**
- § - **De 100 à 900 par pas de 100. 400 = normal, 700 = bold**
- §      **Inherit**

Ø **Exemple :**

§ **font-weight: bolder;**

## 4.5- text-align

Ø Alignement horizontal du

Ø texte Valeurs possibles :

§ left

- Aligné à gauche

§ center

- Centré

§ right

- Aligné à droite

§ justify

- Justifié

§ Inherit

Ø Exemple

§ **text-align: justify;**

## 4.6- text-decoration

### Ø Soulignements divers

### Ø Valeurs possibles

§ **underline**

Texte souligné

§ **overline**

Texte surligné (MSIE)

§ **line-through**

Texte barré

§ **none**

Pas de "décoration", sert à supprimer le soulignement des liens

§ **Inherit**

### Ø Exemples :

§ **text-decoration: none;**

## 4.7- text-transform

### Ø Soulignements divers Valeurs possibles

Ø

§ **Texte en majuscule**

**-uppercase**

§ **Texte en minuscule**

**-lowercase**

§ **Première lettre de chaque mot en majuscule**

**-capitalize**

§ **Pas de transformation**

**-none**

§

**Inherit**

### Ø Exemples :

§ **text-transform: capitalize;**

# 4.8- line-height

Ø Hauteur de l'interligne

Ø Valeurs possibles :

§ *Valeur\_numérique*

- *Valeur suivi de l'unité de mesure*

§ *Pourcentage*

- *Pourcentage relatif à la taille des caractères*

§ **normal**

§ **Inherit**

Ø Exemple :

§ **line-height: 14px;**

## 4.9- vertical-align

- Ø Alignement vertical du texte
- Ø Sert aussi pour d'autres éléments que le texte
- Ø (images...) Valeurs possibles :
  - § baseline : valeur par défaut
  - § middle : centré verticalement
  - § sub : indice
  - § super : exposant
  - § top : alignement partie supérieure de l'élément
  - § bottom : alignement partie inférieure de l'élément
  - § Inherit
- Ø Exemple :
  - § **vertical-align: super;**

## 4.10- Utilisation de la balise <span>

Ø Sert à définir un style pour une sous-partie d'une balise existante.

Ø Exemple :

§ Ce texte est en bleu avec une partie en rouge.

§ Codage :

```
<p style="color: blue">Ce texte est en Arial bleu avec une partie en  
<span style="color: red"> rouge </span></p>.
```

## **5- Les arrière-plans**



## 5- Sommaire

Ø S'utilisent pour mettre une couleur ou une image en arrière-plan.

- § **background-color**
- § **background-image**
- § **background-attachment**
- § **background-repeat**
- § **background-position**
- § **background-size**

# 5.1- background-color

Ø Définit la couleur d'arrière-plan d'un élément HTML.

Ø Valeurs utilisées :

- § transparent (valeur par défaut)
- § *couleur* (nom de couleur ou valeur hexadécimale)
- § Inherit

Ø Exemples :

- § **background-color:**
- § **#ffff00;**      **background-color: yellow;**

## 5.2- background-image

Ø Définit une image d'arrière-plan

Ø Valeurs utilisées :

§ none

- pas d'image, valeur par défaut

§ url

- URL de l'image

§ Inherit

Chemin relatif par rapport à la feuille de style

Ø Exemples :

§ **background-image: url(img/logo.png);**

§ **body {background-image: url(img/fond.png)}**

## 5.3- background-attachment

Ø Définit les caractéristiques d'une image d'arrière-plan.

Ø plan. Valeurs utilisées :

§ scroll

- L'image défile avec le texte L'image est fixée

§ fixed

- 

§ Inherit

Ø Exemples :

**body {background-image: url(fond.gif); background-attachment: fixed}**

## 5.4- background-repeat

Ø Définit le mode de répétition d'une image en arrière-plan.

Ø plan. Valeurs possibles :

§ repeat

- Répétition sur tout l'écran, valeur par défaut) Répétition selon l'axe des X

§ repeat-x

- Répétition selon l'axe des Y

§ repeat-y

- Pas de répétition

§ no-repeat

- 

§ Inherit

Ø Exemples :

**body {background-image: url(fond.jpg); background-repeat: repeat-y}**

## 5.5- background-position

- Ø Définit la position d'une image en arrière-plan. Cela suppose que l'image n'est pas répétée !
- Ø Valeurs possibles :
  - § top
  - § middle
  - § bottom
  - § left
  - § right
  - § center
  - § Pourcentage
- Ø Si deux valeurs : rapport à gauche et au début de page
- Ø Exemples
  - body {background-image: url(fond.jpg); background-repeat: no-repeat; background-position: 50% 50%}

## 5.6- background-size

Ø La propriété CSS **background-size** spécifie la taille de l'image dans l'arrière-plan.

Ø Valeurs possibles :

§ *Dimensions*

- par exemple en "px" ou en "%" : précise la hauteur ou la largeur de l'image.

§ **cover**

- forcera à couvrir toute la surface sans déformer l'image. Quitte à la rogner.

§ **contain**

- forcera l'image à ne pas dépasser l'élément sans la déformer.

Ø Exemples

**body {background-image: url(fond.jpg); background-size: 100%}**

## 6 - Les dimensionnements



## 6- Sommaire

∅ **height**

∅ **width**

∅ **min-height**

∅ **min-width**

∅ **max-height**

∅ **max-width**

# 6.1- height

## Ø Définit la hauteur d'un objet :

- § Image
- § Cellule de tableau
- § Élément de bloc

## Ø Valeurs possibles :

- § *Une hauteur (nombre + unité)*
- § *Un pourcentage*
- § Inherit

## Ø Exemples

- § **height:**
- § **50px**
- height: 50%**

## 6.2- width

Ø Définit la largeur d'un objet :

- § Image
- § Cellule de tableau
- § Élément de bloc

Ø Valeurs possibles :

- § *Une largeur (nombre + unité)*
- § *Un pourcentage*
- § auto
- § Inherit

Ø Exemples

- § **width:**
- § **50px**
- width: 50%**

## 6.3- min-height

Ø Définit la hauteur minimale d'un élément.

Ø Si la hauteur est définie en pourcentage  
cela indique la hauteur minimale.

Ø Valeurs possibles :

- § *Une hauteur (nombre + unité)*
- § *Un pourcentage*
- § **Inherit**

Ø Exemples

§ **img {height: 50%; min-height: 50px}**

## 6.4- min-width

- Ø Définit la largeur minimale d'un élément.
- Ø Si la largeur est définie en pourcentage cela indique la largeur minimale.
- Ø Valeurs possibles :
  - § *Une largeur (nombre + unité)*
  - § *Un pourcentage*
  - § **Inherit**
- Ø Exemples
  - § **img {width: 50%; min-width: 50px}**

## 6.5- max-height et max-width

- ∅ Comme min-height et min-width,
- ∅ Mais concerne la valeur maximale de l'élément.

## 7 - Les bordures



## 7- Sommaire

- Ø **border**
- Ø **border-color**
- Ø **border-style**
- Ø **border-width**
- Ø **border-bottom**
- Ø **border-bottom-color**
- Ø **border-bottom-style**
- Ø **border-bottom-width**
- Ø **border-left**
- Ø **border-left-color**
- Ø **border-left-style**
- Ø **border-left-width**
- Ø **border-right**
- Ø **border-right-color**
- Ø **border-right-style**
- Ø **border-right-width**
- Ø **border-top**
- Ø **border-top-color**
- Ø **border-top-style**
- Ø **border-top-width**

## 7.1- Aspects généraux

- Ø Définit les caractéristiques des bordures des tableaux, des boîtes...
- Ø On peut définir chaque côté séparément
  - § bottom
  - § (dessous)      top
  - § (dessus)
  - § right (Droite)
  - § left (gauche)
- Ø Trois propriétés distinctes à chaque fois :
  - § width (largeur)
  - § style (style)
  - § color (couleur)

## 7.2- border-color

Ø Valables pour toutes les propriétés de type border

Ø Définit la couleur de la bordure

Ø Valeurs possibles :

- § *Une couleur* (nom ou valeur hexadécimale)
- § Inherit

Ø Exemples

- § **border-color: red**
- § **border-color: #ffffaa**

Ø On peut aussi définir chaque bordure séparemement :

- § **border-color: red black blue green**
- § Moyen mnémotechnique : la montre - vous partez de «pile» (haut) et vous tournez dans le sens des aiguilles :
  - haut droit bas gauche

## 7.3- border-style

Ø Valables pour toutes les propriétés de type

Ø border Définit le style de la bordure

Ø Valeurs possibles :

- § **none** (pas de cadre)
- § **dotted** (pointillés)
- § **dashed** (tirets)
- § **solid** (trait plein)
- § **ridge** (trait 3D)
- § **inset** (trait relief rentrant)
- § **outset** (trait relief sortant) Inherit

Nécessite une épaisseur de trait conséquente

Ø Exemples :

- § **border-style: solid**

## 7.4- border-width

Ø Valables pour toutes les propriétés de type border

Ø Définit l'épaisseur de la bordure

Ø Valeurs possibles :

§ Valeur numérique, suivie de l'unité

§ Valeur prédéfinie

- thin (fin)

- medium (moyen)

- thick (épais)

§ Inherit

Ø Exemple

§ **border-width: medium**

## 7.5- border-top, -right, -bottom, -left

- ∅ Border-top, border-right, border-bottom, border-left
- ∅ Pour chacune de ses propriétés, on a la possibilité de commander
  - § Soit les trois styles à la fois
  - § Soit un style à la fois (border-top-style, border-left-color...)
- ∅ Exemples :
  - § **border-top : 2mm dotted**
  - § **#000080 border-right-style: solid**

## 7.6- border-collapse

Ø Définit la façon dont les bordures des cellules d'un tableau sont jointives ou non.

Ø Valeurs possibles :

- § separate : les cellules ont les bordures disjointes (par défaut)
- § collapse : les cellules ont les bordures jointes
- § Inherit

Ø Exemples :

- § **border-collapse: collapse**
- § **border-collapse:**  
**separate**

## 8 - Les marges



## 8- Sommaire

Ø margin

Ø margin-top

Ø margin-right

Ø margin-bottom

Ø margin-left

Ø padding

Ø padding-top

Ø padding-right

Ø padding-bottom

Ø padding-left

## 8.1- margin

- Ø Définit les caractéristiques de la marge extérieure.
- Ø Mêmes remarques que pour les bordures, on peut définir les marges ensemble ou séparément.
- Ø Valeurs possibles :
  - § *valeur numérique + unité*
  - § Inherit
- Ø Exemples :
  - § `margin: 10px 5px 10px 5px; /* H, D, B, G`
  - § `*/ margin: 10px 5px; /* H+B, D+G */`
  - § `margin: 10px; /* H+B+D+G */`

## 8.2- margin-top, -right, -bottom, -left

- Ø Définit la taille des marges extérieure supérieure, droite, inférieure et gauche.
- Ø Valeurs possibles :
  - § *Une valeur numérique + unité*
  - § *Un pourcentage*
  - § **auto**
  - § **Inherit**
- Ø Exemples :
  - § **margin-left: auto; margin-right:auto**
    - Note : cela centre l'élément concerné.

## 8.3- padding

- Ø Définit les caractéristiques de la marge intérieure.
- Ø Mêmes remarques que pour les bordures, on peut définir les marges ensemble ou séparément.
- Ø Valeurs possibles :
  - § *Une valeur numérique + unité*
  - § *Un pourcentage*
  - § **Inherit**
- Ø Exemples :
  - § **padding: 10px 10px 10px 10px;**

## 8.4- padding-top, -right, -bottom, -left

Ø Définit la taille des marges intérieures supérieure, droite, inférieure et gauche.

Ø Valeurs possibles :

- § *Une valeur numérique +*
- § *unité Un pourcentage*
- § Inherit

Ø Exemples :

- § **padding-left : 10px**



## 9 - Les boîtes

# Sommaire

Ø Définition

Ø Les propriétés des boîtes

## 9.1- Définition

- Ø Appelées aussi parfois "calques"
- Ø Définies par :
  - § Leurs dimension
  - § Leurs bordures
  - § Leurs marges
  - § Leurs positionnement dans la page
- Ø Souvent codées avec la balise <div>, mais pas forcément !
  - § Toute balise peut générer une boîte !

## 9.2 Les propriétés des boîtes

### Ø Dimensions :

§ height, width

### Ø Bordures :

§ Toute la série des "border" avec les variantes -color, -style, -width, -top, -left...

### Ø Marges

§ Toute la série des margin et des padding

### Ø Affichage

§ Les propriétés display et visibility

### Ø Positionnement :

§ Sera vu au chapitre suivant

# 9.3 Affichage des boîtes

## Ø Display:

- § **none** : la boîte sera masquée (elle perd son emplacement).
- § **inline** : la boîte sera affichée comme un élément en-ligne
- § **block** : la boîte sera affichée comme un élément bloc.
- § **inline-block** : la boîte sera affichée comme un élément en-ligne avec un comportement bloc (peut être dimensionnée)
- § Autres valeurs :
  - **list-item** crée un bloc principal avec dedans un élément en-ligne pour un item de liste table correspond à 'table'
  - **inline-table** correspond à 'table'
  - **table-row-group** correspond à 'tbody'
  - **table-header-group** correspond à 'thead'
  - **table-footer-group** correspond à 'tfoot'
  - **table-row** correspond à 'tr'
  - **table-column-group** correspond à 'colgroup'
  - **table-column** correspond à 'col'
  - **table-cell** correspond à 'td', 'th'
  - **table-caption** correspond à 'caption'
  -

## Ø Visibility :

- § **hidden** : la boîte sera affichée (elle conserve son emplacement),
- § **visible**: la boîte sera affichée (comme un bloc ou en-ligne suivant son appartenance initiale),
- § **collapse** : utilisée seulement dans les tableaux, dans le cas contraire le comportement est égal à hidden, l'espace normalement occupé par la ligne ou la colonne devient disponible. Si sur une cellule son contenu est caché (équivalent à display: none).

# **10 - Les positionnements**



## 10- Sommaire

- **float**
- **clear**
- **position**
- **left, top, right, bottom z-index**
- **overflow**

# 10.1- float

- Définit l'alignement d'un élément par rapport à un autre élément.
- Correspond à ALIGN en HTML
- Valeurs possibles :
  - left
  - right
  - none
  - Inherit

## Exemple

- `img {float: left}`

## 10.2- clear

- Associé à float
- Force un élément à couler sous un autre élément défini à l'aide de la propriété float.
- Valeurs possibles :

**left** - Le texte se poursuit en dessous de l'élément défini à l'aide de float (float: left)

**right** - La même chose que précédemment avec float: right

**both** - Impose dans chaque cas la poursuite de l'élément

**none** - Pas de règle spécifique

**Inherit**

## 10.3- position

- Permet de positionner un élément sur page de façon absolue ou relative.
- Valeurs possibles :
  - static
    - Comportement normal
  - absolute
    - Par rapport à son parent (si aucun, équivalent à fixed)
  - relative
    - Par rapport à l'élément précédent
  - fixed
    - Par rapport au bord de la fenêtre
  - sticky
    - Positionnement adhérent, mélange de positionnement relatif et de positionnement fixe (voir exemple sur <https://developer.mozilla.org/fr/docs/Web/CSS/position> )
  - Inherit

Exemples :

**position: absolute; left: 10px; top: 100px**

## 10.4- left, top, right, bottom

- Précise la position d'un élément par rapport au bord de la fenêtre ou de l'élément parent.
- S'utilise avec position.
- Valeurs possibles :
  - *Une distance* : valeur numérique + unité
  - *Un pourcentage* : dimension par rapport à l'élément parent.
  - auto
  - Inherit

**Exemples :**

§ **position: absolute; left: 10px; top: 100px**

## 10.5- z-index

- Définit l'ordre de superposition des éléments positionnés à l'aide la propriété position.
- Valeurs possibles :
  - *Un nombre*  
Plus la valeur est élevée, plus l'élément est placé au premier plan.
  - Inherit

Exemple :

- **z-index: 0**
- **z-index: 1**

## 10.6- overflow

- Définit le mode de débordement des éléments notamment ceux dimensionnés avec **height** et **width**
- Valeurs possibles :
  - **visible**
    - Le débordement de l'élément est affiché
  - **hidden**
    - Le débordement est caché
  - **scroll**
    - Des barres de défilement sont affichées
  - **auto**
    - Des barres de défilement sont affichées, si nécessaire
  - **Inherit**

**Exemple :**

**overflow: auto**



## 11 - Les nouveautés du CSS3

# 11- Sommaire

- Réaliser des coins arrondis
- Créer des ombres sur des boites et du texte
- Images multiples dans un même background
- Réaliser un fond en dégradé sans image
- Déclarer une police de caractère non standard avec @font-face
- Animations et transitions d'images ou d'objets
- Pseudo-éléments vs Pseudo-classes
- Les Media Queries

## 11.1- Réaliser des coins arrondis

- Arrondir les coins est sûrement l'un des effets graphiques les plus recherchés par les Webdesigners. Pour cela, on a longtemps utilisé des images avec plus ou moins de bonheur et de facilité pour obtenir l'effet recherché.
- Tous les navigateurs récents permettent à présent d'arrondir les coins de cadres, d'images, de tableaux etc. avec une facilité déconcertante.
- Propriétés utilisées :
  - **border**
  - **border-radius**

## 11.1.1 Réaliser des coins arrondis

### Navigateurs

- Chrome 10
- Safari 3.2
- Firefox 3.6
- Internet Explorer 9+
- Opera 10.5



## 11.1.2 Réaliser des coins arrondis

- Il n'y a normalement plus besoin d'utiliser des préfixes propriétaires tels -moz ou -Webkit pour avoir la bonne restitution de l'arrondissement des coins. Néanmoins, pour assurer la compatibilité avec des versions de Firefox antérieures à la 4, ou pour le Webkit de vieux smartphones, il vaut mieux encore les laisser.
- La propriété **border-radius** peut accepter 4 valeurs pour l'arrondissement de chaque coins. La 1ère valeur correspond au coin haut gauche, puis on tourne dans le sens des aiguilles d'une montre. On peut n'en indiquer que 2, qui correspondront aux coins opposés (voir l'exemple ci-dessous), ou une seule pour un même arrondis sur les 4 coins.

## 11.1.3 Réaliser des coins arrondis

```
<div id="coin">  
  <p>  
    Lorem ipsum dolor sit amet, consectetur adipiscing  
    elit. Mauris vulputate laoreet urna. Integer magna.  
    Donec facilisis lectus sed quam. Curabitur sit amet  
    lacus id lacus facilisis venenatis.  
  </p>  
</div>
```

```
#coin {  
  background-color: #E4EFFF;  
  border: 1px solid #9FC6FF;  
  padding: 5px;  
  /*arrondir les coins en haut à gauche et en bas à  
  droite*/  
  -moz-border-radius: 10px 0;  
  -Webkit-border-radius: 10px 0;  
  border-radius: 10px 0;  
}
```

## 11.1.4 Arrondir les coins d'une image

Pour arrondir les coins d'une image, même principe :

```
<p>

</p>
```

```
/*sans bordure (1ère photo)*/
.arrondie {
-moz-border-radius: 7px;
-Webkit-border-radius: 7px; border-radius: 7px;
}
```

## 11.1.5 Arrondir les coins d'une image

Cependant, cette technique ne fonctionne pas avec Opera 11.5 par exemple. On peut alors ruser en déclarant une bordure qui donnera l'impression de masquer les coins. Cela ne sera néanmoins valable que pour de petits rayons d'arrondissement car ce n'est qu'une pure illusion d'optique.

```
/*avec bordure (2ème photo) */ .arrondie2 {  
border: 2px solid black;  
-moz-border-radius: 7px;  
-Webkit-border-radius: 7px; border-radius: 7px;  
}
```

## 11.2 Créer des ombres sur des boites et du texte

- Ombrer des boites ou du texte sans images c'est possible ! Plus besoin d'avoir recours à des logiciels « photoshoppeurs » pour faire et refaire : un peu de code CSS suffit.
- Voici les explications, illustrées de quelques exemples, de réalisations d'ombres en CSS.
- Sommaire :
  - **Ombrer des boites**
  - **Ombrer du texte**
  - **Lectures complémentaires**

## 11.2.1 Créer des ombres sur des boites et du texte

### Navigateurs

- **Chrome 10**
- **Safari 3.2**
- **Firefox 3.6**
- **Internet Explorer 9+**

**Opera 10.6**



## 11.2.2 Ombrer des boîtes

**La propriété box-shadow doit recevoir plusieurs valeurs (les deux premières étant obligatoires) :**

- Une valeur de décalage horizontal
- Une valeur de décalage vertical
- Une valeur pour l'effet flou
- Une valeur l'étendue de l'ombre
- La couleur de l'ombre

## 11.2.2.1 Ombrer des boîtes

```
<div id="ombreout">  
<p>  
Lorem ipsum dolor sit amet, consectetur adipiscing  
elit. Mauris vulputate laoreet urna. Integer magna.  
Donec facilisis lectus sed quam. Curabitur sit amet  
lacus id lacus facilisis venenatis.  
</p>  
</div>
```

```
#ombreout {  
background-color: #C0C0C0;  
border: 1px solid gray;  
padding: 5px;  
box-shadow: 2px 2px 10px gray;  
-moz-box-shadow: 2px 2px 10px gray;  
-Webkit-box-shadow: 2px 2px 10px gray;  
}
```

## 11.2.2 Ombrer des boîtes

```
<div id="ombrein">  
<p>  
Lorem ipsum dolor sit amet, consectetur adipiscing  
elit. Mauris vulputate laoreet urna. Integer magna.  
Donec facilisis lectus sed quam. Curabitur sit amet  
lacus id lacus facilisis venenatis.  
</p>  
</div>
```

```
#ombrein {  
background-color: #C0C0C0;  
border: 1px solid gray;  
padding: 5px;  
box-shadow: 2px 2px 10px gray inset;  
-moz-box-shadow: 2px 2px 10px gray inset;  
-Webkit-box-shadow: 2px 2px 10px gray inset;  
}
```

## 11.2.3 Ombrer du texte

Contrairement à box-shadow, text-shadow n'est pas implémenté dans Internet Explorer...

Sinon, aucun besoin de préfixes propriétaires et, hormis pour l'étendue qui n'existe pas, même principe que box-shadow pour les valeurs à lui attribuer.

```
<h1>Titre ombré</h1>
```

```
h1 {  
    text-shadow: 1px 1px 2px gray;  
}
```

### 11.2.3.1 Ombrer du texte

Quant à l'effet "gravure" que l'on rencontre de plus en plus, il utilise le même procédé mais avec une couleur plus claire que la couleur de fond.

```
<h1>Titre ombré</h1>
```

```
h1 {  
    text-shadow: 1px 1px 2px white;  
}
```

## 11.3 Images multiples dans un même background

La propriété `background-image` n'acceptait, en CSS 2.1, qu'une seule image. Pour pouvoir en mettre plusieurs à différents endroits d'une page, il fallait déclarer plusieurs `div` ayant chacun sa propre image de fond. Lourd...

Les CSS3 permettent des `background` multiples, et maintenant que même Internet Explorer (version 9) l'implémente, pourquoi s'en priver ?

# Images multiples dans un même background

## ∅ Navigateurs

- § Chrome 4+, Safari
- § 4+ Firefox 3.6+
- § Internet Explorer 9+
- § Opéra 10.10+



## 11.3.1 Images multiples dans un même background

- Ø Les CSS3 apportent une vraie simplification et une souplesse à toute épreuve avec les background multiples.
- Ø La syntaxe est de plus extrêmement simple : elle est identique au **background de CSS 2.1**, chaque nouveau fond devant être simplement séparé par une virgule.

## 11.3.2 Images multiples dans un même background

Ø Ainsi, pour reproduire le même effet que ce rendu d'images multiples à l'aide de div, il suffira d'indiquer ceci :

```
<div id="contenu">
  <h1>Les quatre coins</h1>
  <p> [bla bla]</p>
</div>
```

```
#contenu {
  background: url(images/houx2.gif) no-repeat left top,
  url(images/scream.jpg) no-repeat right top,
  url(images/scream.jpg) no-repeat left bottom,
  url(images/scream.jpg) no-repeat right bottom;
}
```

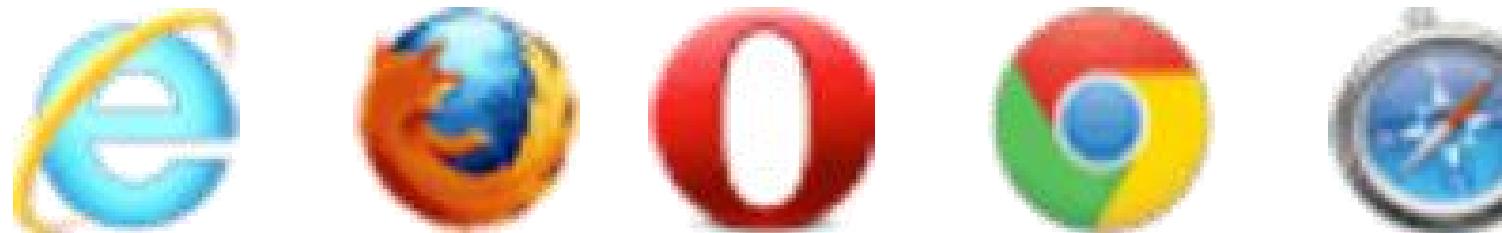
# 11.4 Réaliser un fond en dégradé sans image

- Ø Avant les CSS3, pour réaliser de jolis boutons ou tout autre fond de couleur en dégradé, il convenait de réaliser une image que l'on déclarait en image de fond à l'aide de `background-image`.
- Ø On peut maintenant s'en passer grâce à de nouvelles valeurs de `background` : `linear-gradient` et `radial-gradient`.
- Ø Sommaire
  - § Principe de base
    - Dégradés linéaires
    - Dégradé radial
  - § Réalisation d'un bouton
  - § Filtre pour Internet Explorer
  - § Lectures complémentaires

## 11.4.1 Réaliser un fond en dégradé sans image

### ∅ Navigateurs

- § **Chrome 3, Safari**
- § **5.0 Firefox 3.6**
- § **Internet Explorer**
- § **10 Opera 11.6**



## 11.4.2 Réaliser un fond en dégradé sans image

Ø Le principe de base est très simple :

- § On définit le type de dégradé voulu : linéaire (linear-gradient) ou radial (radial-gradient) ;
- § On indique les couleurs de départ et d'arrivée et, éventuellement, leurs proportions de mélange.

Ø Attention : les codes CSS indiqués ci-après ne tiennent compte que ponctuellement des préfixes propriétaires à rajouter selon le navigateur (-moz- pour Firefox, -Webkit- pour Safari et Chrome, et -o- pour Opera). Il convient néanmoins de les indiquer systématiquement si l'on veut assurer une rétrocompatibilité.

## 11.4.3 Réaliser un fond en dégradé sans image

```
<div class="degrade">
<p>
    Lorem ipsum dolor sit amet, consectetur adipiscing
    elit. Mauris vulputate laoreet urna. Integer magna.
    Donec facilisis lectus sed quam. Curabitur sit amet
    lacus id lacus facilisis venenatis.
</p>
</div>
```

```
.degrade {
    width: 250px;
    color: red;
    background-image: linear-gradient(white, black);
}
```

## 11.4.4 Réaliser un fond en dégradé sans image

### Ø Dégradé de la gauche vers la droite

```
.degrade {  
    width: 250px;  
    color: red;  
    background-image: linear-gradient(  
        to right,  
        white,  
        black);  
}
```

## 11.4.5 Réaliser un fond en dégradé sans image

Ø Dégradé de la gauche vers la droite avec limite du fondu

```
.degrade {  
    width: 250px;  
    color: red;  
    background-image: linear-gradient(  
        to right,  
        white 10%,  
        black 90%);  
}
```

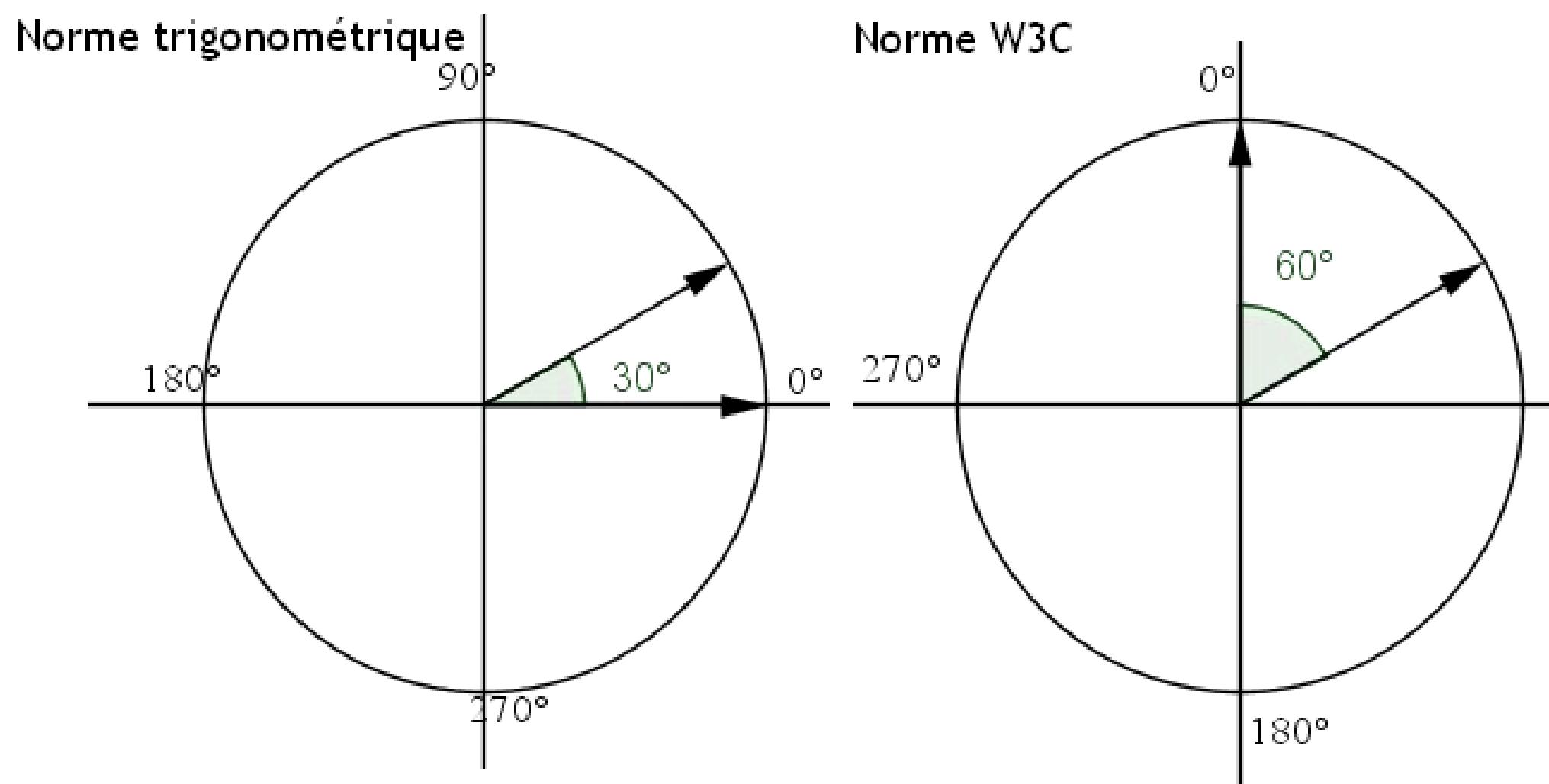
## 11.4.6 Réaliser un fond en dégradé sans image

Ø Dégradé suivant un angle de 60 degrés

```
.degrade {  
    width: 250px;  
    color: red;  
    background-image: linear-gradient(  
        60deg,  
        white,  
        black);  
}
```

## 11.4.7 Réaliser un fond en dégradé sans image

Ø Attention : les navigateurs utilisant encore des versions préfixées et ceux implémentant les dernières spécifications du W3C ne mesurent pas les angles de la même façon. Les anciens suivent le principe du cercle trigonométrique alors que le W3C prend l'origine de l'angle à la verticale et tourne dans le sens des aiguilles d'une montre.



## 11.4.8 Réaliser un fond en dégradé sans image

- Ø Ainsi une même orientation aura pour mesure 30 degrés en version préfixée et 60 degrés en version normalisée W3C.
- Ø Le code devra donc être pour un maximum de compatibilité :

```
.degrade {  
    width: 250px;  
    color: red;  
    background-image:-moz-linear-gradient(30deg, white,  
black);  
    background-image:-webkit-linear-gradient  
white, black);  
    background-image:-o-linear-gradientdeg, white,  
black);  
    background-image:linear-gradient(60deg, white,  
black);  
}
```

## 11.4.9 Réaliser un fond en dégradé sans image

### Ø Dégradé axial

```
.degrade {  
    width: 250px;  
    color: red;  
    background-image: radial-gradient(  
        white,  
        black);  
}
```

## 11.4.10 Réaliser un bouton

```
.bouton {  
width: 150px;  
height: 50px;  
background: black;  
color: white;  
border-radius: 10px;  
background: linear-gradient(  
#828c95,  
#000000);  
}  
.bouton:hover {  
background: linear-gradient(  
#ccc6c3 0%,  
#828c95 24%,  
#000000 41%,  
#7f7f7f 100%);  
}
```

## 11.5 Déclarer une police de caractère non standard avec @font-face

- Longtemps, longtemps, le Webmestre s'est lamenté de ne pouvoir utiliser qu'un petit nombre de fontes pour ses sites Web. Arial, Helvetica, times new roman... c'était d'un triste ! Ou alors il fallait faire des images. Et mettre du texte en image ce n'est tout de même pas très logique, sans compter les problèmes d'accessibilité qui peuvent en découler.
- Grâce à @font-face on peut à présent utiliser des polices de caractère exotiques et laisser cours à son inventivité.

## 11.5.1 Déclarer une police de caractère non standard avec @font-face

### Navigateurs

- Chrome 10
- Safari 3.2
- Firefox 3.6
- Internet Explorer 9+ Opera 10.6



## 11.5.2 Déclarer une police de caractère non standard avec @font-face

Le sélecteur `@font-face` existe depuis très longtemps : depuis les CSS 2.0. Mais seul Internet Explorer l'implémentait depuis sa version 4 (oui, oui, vous avez bien lu !) et avec un format de police propriétaire peu pratique. Cela n'a donc jamais pris. Heureusement, il y a maintenant des moyens simples pour surmonter cet écueil.

La police de caractère, ou plutôt les différents formats de la police de caractère, doit être placée sur le serveur. Comme pour une image de fond, elle sera chargée au moment de l'appel de la feuille de style dans laquelle est déclarée `@font-face`.

### 11.5.3 Déclarer une police de caractère non standard avec @font-face

Il y aura donc les mêmes contraintes techniques et éthiques qu'avec une image :

1. il faut faire attention au poids pour éviter des lenteurs de chargement et donc les problèmes de performances ;
2. les droits d'auteur sont peut-être encore plus féroces que pour les images, il faut donc n'utiliser que des polices dont on est en accord avec la licence.
  - Pour l'exemple, nous allons utiliser la police de caractère **quadranta**, gratuite, disponible sur le site [Dafont](#).

## **11.5.4.1 Étape 1 : obtenir tous les formats nécessaires**

Quadranta est téléchargeable sous le format .otf (OpenType Font). Ce format est reconnu par Firefox, Chrome, Safari, Opéra. En revanche il n'est reconnu ni par MSIE, ni par certains navigateurs de smartphones.

Pour être complet, il donc faut aussi avoir cette police aux formats .eot (Embedded OpenType, pour MSIE) et .svg. Existent aussi les formats .ttf (TrueType Font, très répandu) et .woff (Web Open Format Font) qui est le petit format qui monte, qui monte.

## 11.5.4.2 Étape 1 : obtenir tous les formats nécessaires

Où et comment nous allons pouvoir récupérer tous ces formats ? [FontSquirrel](#) est un générateur de font-face va nous fournir tous les formats nécessaires.

Nous disposons au final les fichiers suivants (ainsi qu'un .ttf qui ne nous servira pas) :

**quadranta.otf** : 20 ko

**quadranta.woff** : 26 ko

**quandrata.eot** : 35 ko

**quadranta.svg** : 58 ko

- Pour des raisons de performance il faudra que l'ordre de déclaration des différents formats soit identique (du plus léger au plus lourd).
- Vous pouvez aussi convertir vos polices sur [font2Web](#)

# Étape 2 : coller le code dans la feuille de style

FontSquirrel fournit aussi le code CSS. Après avoir reclassé les formats par poids croissant et remplacé le .ttf par .otf, il est de ce type :

```
@font-face {  
    font-family: 'quadranta';  
    src: url('quadranta.eot');  
    src: url('quadranta.otf') format('truetype'),  
        url('quadranta.woff') format('woff'),  
        url('quadranta.eot?#iefix') format('embedded-opentype'),  
        url('quadranta.svg#QuadrantaBold') format('svg');  
    font-weight: normal;  
    font-style: normal;  
}
```

### 11.5.4.3 Étape 2 : coller le code dans la feuille de style

Le nom déclaré par font-family sera celui à utiliser dans le reste de la feuille de style aux endroits souhaités.

L'ancre #QuadrantaBold déclarée pour le format .svg est nécessaire pour que les navigateurs retrouvent leurs petits. C'est en effet l'id inscrit dans le fichier svg et qui indique le chemin à suivre. Il ne faut donc pas modifier cette déclaration.

## 11.5.4.4 Étape 2 : coller le code dans la feuille de style

Restent les déclarations .eot : pourquoi 2, pourquoi différentes ?

MSIE a beau être le navigateur le plus ancien à implémenter @font-face, il fallait quand même bien qu'il complique un peu les choses, sinon ce ne serait pas drôle... Il y a une question de performance (sinon MSIE charge tous les formats, même ceux qu'il ne gère pas) et ensuite les nouveautés d'IE9, qui comprend d'ailleurs le format .woff plus léger et qu'on préférera lui faire ingurgiter...

- En fait, cette double déclaration peut être remplacée par une seule :

`url('quadranta.eot?') format ('eot')`

- Tout est dans le ? et la déclaration de `format('eot')`. Et si vous voulez comprendre le pourquoi du comment, courez sur **Typographisme** qui explique tout ça très bien.

## 11.5.4.5 Étape 2 : coller le code dans la feuille de style

Ø Le format .eot devant être déclaré en 1er, le code CSS final sera donc :

```
@font-face {  
    font-family: 'quadranta';  
    src: url('quadranta.eot?') format('eot'),  
        url('quadranta.otf') format('truetype'),  
        url('quadranta.woff') format('woff'),  
        url('quadranta.svg#QuadrantaBold') format('svg');  
    font-weight: normal;  
    font-style: normal;  
}  
h4 {  
    font-family: quadranta, sans-serif;  
    font-size: 3em;  
}
```

## 11.6 Animations et transitions d'images ou d'objets

- Ø L'ère des gifs animés est morte, celle de flash est en déclin. La faute à... ? Aux CSS3 et à HTML5 qui apportent leur flopée de nouveaux outils pour animer nos sites !
- Ø animation et transition permettent d'apporter du mouvement aux pages Web en quelques lignes de code
- Ø Propriétés utilisées :
  - § animation
  - § transition
- Ø Sommaire
  - § Réaliser une transition
  - § Réaliser une animation
  - § Lectures complémentaires

## 11.6.1 Réaliser une transition

### ∅ Navigateurs

- § **Chrome 4**
- § **Safari 3.1**
- § **Firefox 4**
- § **Internet Explorer**
- § **10 Opera 10.5**



## 11.6.1 Réaliser une transition

- Ø Un changement de couleur de fond au survol de la souris, ou au focus d'un lien, se fait par défaut de façon binaire : on passe du tout au rien. À l'aide de transition, on peut faire en sorte que ce comportement se fasse de façon progressive.
- Ø L'exemple ci-après permet de modifier la transparence de la couleur déclarée en rgba en la faisant passer de 60% (une opacité de 0,4) à une opacité totale. Cette transition s'effectuera en 2 secondes de façon constante (accélération linéaire).
- Ø Pour les navigateurs plus anciens n'interprétant pas les couleurs rgba, on prendra soin de déclarer, juste avant, le background avec des codes hexadécimaux classiques.

```
<p id="doigt"><a href="#doigt">On ne montre pas du doigt !</a></p>
```

```
#doigt a {  
    padding: 10px 10px 10px 77px;  
    background:#FAC27E url(images/scream.jpg) no-repeat 5px 10px;  
background: rgba(255,140,0,0.4)  
url(images/scream.jpg) no-repeat 5px 10px;  
border: 1px solid #FF8C00;  
border-radius: 5px;  
}  
p#doigt a:hover, p#doigt a:focus {  
    color: black;  
    background-color:#ff8c00;  
background-color: rgba(255,140,0,1);  
transition-property: background-color;  
transition-duration: 2s;  
transition-timing-function: linear;  
}
```

## 11.6.1 Réaliser une transition

Ø De façon raccourcie, on peut déclarer simplement :

```
transition: background-color 2s linear;
```

Ø Utilisez les préfixes **-moz** (firefox), **-Webkit** (chrome et safari) , **-o** (opéra) et **-ms** pour Internet Explorer devant la propriété transition pour des raisons de compatibilité.

```
-webkit-transition: background-color 2s linear;  
-moz-transition: background-color 2s linear;  
-ms-transition: background-color 2s linear;  
-o-transition: background-color 2s linear;  
transition: background-color 2s linear;
```

## 11.6.2 Réaliser une animation

### ∅ Navigateurs

- § **Chrome 4**
- § **Safari 4**
- § **Firefox 5**
- § **Internet Explorer**
- § **10 Opera 12**



## 11.6.2 Réaliser une animation

Ø Une animation se déclare en deux étapes dont l'ordre n'a pas d'importance :

- § La première consiste à indiquer, au minimum, le nom que l'on donne à l'animation (`animation-name`), sa durée (`animation-duration`) et le nombre de boucles (`animation-iteration-count`) ;
- § pour la seconde, déclarée à l'aide d'une règle intitulée de `@keyframes` à laquelle on accole le nom de l'animation, on notifie les effets voulus aux moments voulus, ces derniers étant indiqués en pourcentage du temps total déclaré.

Ø L'animation ci-après, qui se déclenche toujours au focus ou au survol de la souris, consiste à faire aller et venir la petite main devant le lien. Quant à la modification de transparence du background, elle va se payer le luxe de suivre le déplacement de l'image.

```
/* première étape */
p#doigt a:hover, p#doigt a:focus {
background-color: #ff8c00; /*pour les vieux navigateurs*/
color: black;
animation-name: doigt; /* nom de l'animation */
animation-duration: 4s; /* temps de l'animation (4s) */
animation-iteration-count: infinite; /* nombre de boucles
(infinie)*/
animation-timing-function: linear; /* accélération de l'animation
(constant) */
}

/* deuxième étape */
@keyframes doigt {
0% {background: rgba(255,140,0,0.4) url(/images/main.png) no-repeat
5px 10px; }
50% {background: rgba(255,140,0,1) url(/images/main.png) no-repeat
5px 10px; }
100% {background: rgba(255,140,0,0.4) url(/images/main.png) no-
repeat 5px 10px; }
}
```

## 11.6.2 Réaliser une animation

Ø La version raccourcie de la règle pour animation est la suivante :

**animation: doigt 4s infinite linear;**

Mais là encore, il faudra les préfixes propriétaires, ceux du @keyframes s'insérant entre le @ et keyframe.

Exemple : @-moz-keyframes.

## 11.6.3 Rotation

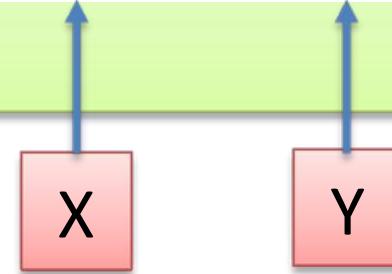
- Ø La fonction CSS rotate de la propriété CSS transform permet d'effectuer une rotation à un élément HTML.
- Ø La fonction rotate CSS peut prendre accepte comme paramètre:
  - § angle, nombre suivit par deg, grad, rad, turn. Peut être négatif. Dans ce cas, le sens de la rotation se fera vers la gauche au lieu de la droite.
  - § Si l'angle est à 0, c'est la position initiale.

```
identifiant
{
    transform: rotate(20deg);
}
```

## 11.6.4 Translation

- Ø La fonction CSS translateX de la propriété transform permet de déplacer horizontalement un élément HTML par rapport à sa position d'origine.
- Ø La fonction translateY : pour un déplacement verticale.
- Ø La fonction translate : pour un déplacement horizontale et verticale.

```
identifiant
{
    transform: translate(20px, 10px);
}
```

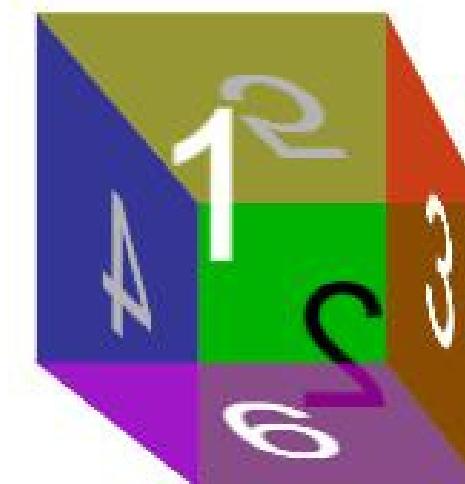


## 11.6.5 Perspective

- La fonction perspective de la propriété transform détermine la distance entre le plan d'équation  $z = 0$  et la position de l'utilisateur afin de donner une perspective aux objets positionnés dans l'espace 3D.
- Par défaut, le point de fuite est placé au centre de l'élément mais il peut être déplacé grâce à la propriété perspective-origin.

```
#red
{
    transform: perspective(600px) rotate(20deg);
}
```

<https://developer.mozilla.org/fr/docs/Web/CSS/perspective>



## 11.7 Pseudo-éléments vs Pseudo-classes

- Une pseudo-classe est une manière de cibler un élément sans ajouter une classe (manuellement ou par l'intermédiaire de JavaScript).

*Exemple : à la place de `:first-child`, on aurait simplement pu ajouter la classe `first` sur le premier élément que l'on souhaitait cibler.*

- Un pseudo-élément est une chose que l'on n'aurait pas pu cibler autrement qu'en rajoutant un élément.

*Exemple : à la place de `:first-letter`, on aurait dû ajouter un élément `span` sur la première lettre de l'élément ciblé.*

- CSS3 propose une nouvelle syntaxe pour les pseudo-éléments. En lieu est place du « `:` » il faudra écrire « `::` » (deux fois deux-points). La syntaxe d'une pseudo-classe reste inchangée.

## 11.7.1 Les pseudo-éléments du CSS3

| Pseudo-élément       | Définition   |
|----------------------|--|
| :nth-child(N)        | cible le Nième élément enfant  |
| :nth-last-child(N)   | cible le Nième dernier élément enfant                                      |
| :nth-of-type(N)      | cible le Nième élément d'un type donné                                     |
| :nth-last-of-type(N) | cible le Nième dernier élément d'un type donné                             |
| :last-child          | cible le dernier élément enfant  |
| :first-of-type       | cible le premier element d'un type donné (équivaut à nth-of-type(1))       |
| :last-of-type        | cible le dernier element d'un type donnée (équivaut à nth-last-of-type(1)) |
| :only-child          | cible un élément s'il est l'unique enfant                                  |
| :only-of-type        | cible un élément s'il est l'unique enfant ce de type                       |
| :root                | cible l'élément racine du document   |

## 11.7.1 Les pseudo-éléments du CSS3

| Pseudo-<br>élément          | Définition   |
|-----------------------------|--|
| <code>:empty</code>         | cible un élément sans enfant   |
| <code>:target</code>        | cible un élément pointé par son identifiant dans l'URL (ancre)             |
| <code>:enabled</code>       | cible les éléments de l'interface qui sont activés                         |
| <code>:disabled</code>      | cible les éléments de l'interface qui sont désactivés                      |
| <code>:checked</code>       | cible les éléments cochés (bouton radio ou case à cocher)                  |
| <code>:required</code>      | cible les éléments de formulaire définis comme requis (attribut required)  |
| <code>:valid</code>         | cible les éléments de formulaire respectant la valeur attendue             |
| <code>:invalid</code>       | cible les éléments ne respectant pas la valeur attendue                    |
| <code>:not(selector)</code> | cible les éléments qui ne correspondent pas au sélecteur entre parenthèses |

## 11.7.2 Les nouveaux pseudo-éléments de CSS3

- Utiliser les mêmes que CSS2 avec un « :: » devant ;
- **::selection : permet de styler du texte sélectionné (couleur de fond, couleur de texte).**

## 11.7.4 Exemples :

- Validation
- Tableau
- Rotation
- Translation

## 11.7.5.1 Validation

- focus devant invalid pour que le champ ne se mette en erreur que lorsqu'il a le focus.

```
input:required
{
    background:url(img/asterisk.png) 98% center no-repeat;
}

input:valid
{
background:url(img/valid.png) 98% center no-repeat;
}

input:focus:invalid
{
background:url(img/error.png) 98% center no-repeat;
}
```

```
input:required
{
    background:url(img/asterisk.png) 98% center no-repeat;
}

input:focus:valid {
    border-color:#56C93F;
    box-shadow:0px 0px 5px #56C93F;
background-image:none;
}

input:focus:invalid {
    border-color:#FF0000;
    box-shadow:0px 0px 5px #FF0000;
background-image:none;
}

input:valid {
    background-image:none;
}
```

## 11.7.5.2 Tableau

- Définir des couleurs de lignes différentes dans un tableau.
- even et odd indique respectivement les lignes paires et impaires. On peut aussi mettre 2n et 2n+1.

```
tbody tr:nth-child(even)
{
    background-color:#FFFFFF;
}

tbody tr:nth-child(odd)
{
    background-color:silver;
}
```

# Autres exemples

```
q::before {  
    content: "« ";  
    color: blue;  
}  
q::after {  
    content: " »";  
    color: red;  
}  
q::selection {  
    color: red;  
    background-color: green;  
}
```

<q>Quelques guillemets</q>, dit-il, <q>sont mieux que pas du tout</q>

« Quelques **guillemets** », dit-il, « sont mieux que pas du tout »

## 11.8 Les Media Queries

Avec l'avènement des smartphones ou autre tablette, la multiplication des tailles et des résolutions d'écran, il devient de plus en plus aléatoire de ne réaliser qu'une seule disposition graphique de sa page Web...

Comment en effet visualiser correctement sur un smartphone une page qui aurait été pensée pour une résolution minimale de 1900px ? (ce qui est *très mal*, d'ailleurs...).

- *Les Media Queries, permettent de cibler différents cas et ainsi d'adapter la restitution de sa page html à différentes caractéristiques des terminaux.*

## 11.8.1 Les Media Queries

Les Media Queries permettent donc de cibler :

- Le type de média
- La taille de l'écran
- La taille de la fenêtre
- La résolution
- Le nombre de couleurs
- L'orientation
- ...

Pour en expliquer le principe, on se limitera à deux exemples :

- **La largeur de la fenêtre** avec un contrôle du width ;
- **La largeur de l'écran du terminal** avec le contrôle de device-width (ce n'est pas la même chose que width tout seul).

## 11.8.2 Les Media Queries

### Navigateurs

- Mozilla Firefox : 3.5+
- Internet Explorer : 9+
- Google Chrome : 5+
- Opera : 10.5+, Opera Mobile : 10+, Opera Mini : 5+  
Apple Safari : 4+ et iOS (mobile) 3.2+
- Android : 2.1+
- WebKit en général



### 11.8.3 Cibler la largeur de la fenêtre

Petit exemple très visuel : changer la couleur de fond initialement blanche de la page lorsque la taille de la fenêtre du navigateur passe au dessous des 800 pixels de large.

Très simplement, alors que la couleur de fond (`background-color`) du `body` est déclarée blanche, on déclarera une couleur noire pour tout écran (`screen`) ayant une taille maximum (`max-width`) de 800px, dans une déclaration `@media {}`.

## 11.8.4 Cibler la largeur de la fenêtre

Avec un smartphone, dont on ne peut modifier la taille de la fenêtre, le résultat sera différent selon son "viewport" (partie visible de l'écran). Et de ce côté là, les constructeurs s'en donnent à cœur joie pour multiplier les configurations.

Un android a un viewport de 800px, mais un iphone4 en a un de 980px. Dans le 1er cas, le fond sera donc noir alors que dans le second il sera blanc...

```
body {  
    background-color: white;  
    color: black;  
}  
  
@media only screen and (max-width:800px) {  
    body {  
        background-color: black;  
        color: white;  
    }  
}
```

## 11.8.5 Cibler la largeur de l'écran

Avec `device-width`, on ne ciblera plus la largeur de la fenêtre, mais la largeur de l'écran.  
Intéressant pour les smartphones...

Reprendons donc le même principe que ci-dessus, mais en déclarant un `max-device-width:480px` à la place du `max-width:800px`.

```
body {  
    background-color: white;  
    color: black;  
}  
@media only screen and (max-device-width:800px) {  
    body {  
        background-color: black;  
        color: white;  
    }  
}
```

## 11.8.7 Appliquer à la taille des caractères

Une application sans doute plus intéressante pour la dimension des polices de caractère souvent trop petite et donc illisible sans zoom sur un écran de téléphone portable. Couplée à une déclaration sur la taille du viewport dans les meta de l'en-tête de la page, la redéfinition de la taille des caractères fera des miracles.

```
<meta name="viewport" content="width=device-width" />
```

```
body {  
    font-size: 100%;  
}  
@media only screen and (max-device-width: 980px) {  
    body {  
        font-size: 120%;  
    }  
}
```

## 11.8.8 L'attribut media

L'attribut media peut prendre (depuis CSS2) les valeurs suivantes :

**screen** Écrans

**handheld** Périphériques mobiles ou de petite taille **print** Impression

**aural (CSS 2.0) / speech (CSS 2.1)** Synthèses vocales

**braille** Plages braille

**embossed** Imprimantes braille

**projection** Projecteurs (ou présentations avec slides)

**tty** Terminal / police à pas fixe

**tv** Téléviseur

**all** Tous les précédents

## 11.8.9 Syntaxe des Media Queries CSS3

- Les opérateurs logiques peuvent être :
  - **and "et"**,
  - **only "uniquement"**
  - **not "non"**.

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

ou

```
@media screen and (max-width: 640px) {  
    .bloc {  
        display: block;  
        clear: both;  
    }  
}
```

## 11.8.10 Fonctionnalités

La plupart des critères (ou fonctionnalités) peuvent être préfixés par min- et max- lorsqu'elles acceptent des valeurs numériques pour définir des valeurs minimales ou maximales à respecter.

| Valeur                     | Définition  |
|----------------------------|---|
| <b>color</b>               | support de la couleur (bits/pixel)                          |
| <b>color-index</b>         | périphérique utilisant une table de couleurs indexées       |
| <b>aspect-ratio</b>        | ratio du périphérique de sortie (par exemple 16/9)          |
| <b>device-aspect-ratio</b> | ratio de la zone d'affichage                                |
| <b>device-height</b>       | dimension en hauteur du périphérique                        |
| <b>device-width</b>        | dimension en largeur du périphérique                        |
| <b>grid</b>                | périphérique bitmap ou grille (ex : lcd)                    |
| <b>height</b>              | dimension en hauteur de la zone d'affichage                 |
| <b>monochrome</b>          | périphérique monochrome ou niveaux de gris (bits/pixel)     |
| <b>orientation</b>         | orientation du périphérique (portait ou landscape)          |
| <b>resolution</b>          | résolution du périphérique (en dpi, dppx, ou dpcm)          |
| <b>scan</b>                | type de balayage des téléviseurs (progressive ou interlace) |
| <b>width</b>               | dimension en largeur de la zone d'affichage                 |

## 11.8.11 Fonctionnalités

Les dimensions pourront être évaluées avec des unités (px, em). Les ratio avec des fractions (entier/entier). Une résolution sera définie en dpi (points par pouce) ou en dpcm (points par centimètres).

Certaines de ces propriétés peuvent être testées d'une façon raccourcie sans valeur, par exemple (color) qui sera équivalent à (min-color: 1) ou considérée comme vraie pour une valeur différente de 0.

La fonctionnalité monochrome n'est pas uniquement booléenne avec la syntaxe raccourcie (monochrome), on peut aussi considérer un nombre de niveaux de gris, par exemple (min-monochrome: 2) pour 2 bits par pixel.

## 11.8.12 Au service des mobiles

Dans la majorité des cas, on utilise les *media Queries* pour produire des améliorations spécifiques à l'affichage sur les mobiles, qui sont directement concernés par des critères sur les dimensions de l'écran (en terme de résolution et d'espace disponible) et sur l'utilisation tactile. Les périphériques sous iOS (iPhone, iPod, iPad...) supportent relativement bien les *media Queries*.

## 11.8.13 Au service des mobiles

Ainsi on retrouvera le plus fréquemment des règles pour :

- agrandir la taille du texte
- agrandir la taille des contrôles et zones cliquables (pour une utilisation au doigt)
- faire passer le contenu sur une seule colonne
- masquer ou afficher des éléments spécifiques
- ajuster les dimensions et marges

## 11.8.14 Au service des mobiles

La fonctionnalité orientation a été introduite pour des périphériques pouvant être orientés, par exemple l'iPad. On peut alors déclarer une feuille de style spécifique pour ajuster l'affichage :

```
<link rel="stylesheet"  
media="(orientation:portrait)"  
href="portrait.css">  
  
<link rel="stylesheet"  
media="(orientation:landscape)"  
href="paysage.css">
```

Certains navigateurs classiques l'implémentent également en version bureau et calculent le ratio hauteur / largeur pour simuler un affichage "orienté".

## 11.8.14 Au service des mobiles

Autres exemples :

- Impression sur un support plus large que 5 pouces :

`print and (min-width: 5in)`

- Ecrans possédant un ratio 16/9 ou 16/10e :

`tv, (device-aspect-ratio: 16/9), (device-aspect-ratio: 16/10)`