



République Tunisienne  
Ministère de l'Enseignement Supérieur,  
de la Recherche Scientifique



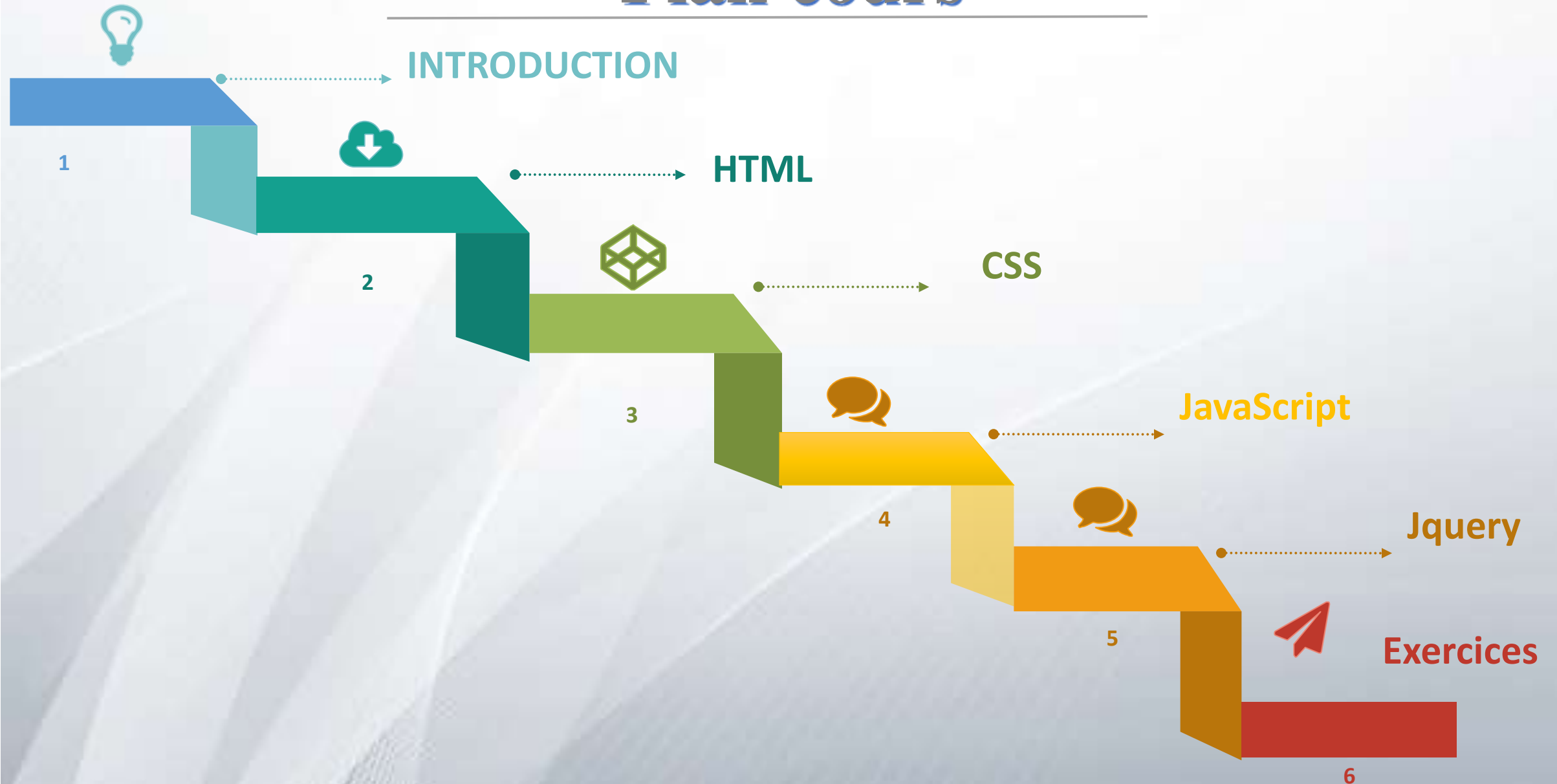
# Technologies et programmation web CSS

**Njeh Maissa**

**Audiences: DLSI-ADBD**

**Année universitaire : 2023/2024**

# Plan cours



# CSS

# 4 CSS : Cascading Style Sheets

## CSS signifie Feuilles de Style en Cascade

- CSS est le langage que nous utilisons pour **styler** un document HTML.
- CSS décrit comment les éléments HTML doivent être affichés.
- C'est en CSS que l'on dira : "mes titres sont en rouge et sont soulignés, mon texte est dans la police arial, mon menu a un fond blanc.... "
- On peut écrire du code **CSS** a 3 endroit différents, selon ce qu'on préfère:

○ Méthode A : dans un fichier .css

○ Méthode B: dans l'en-tete du fichier html

○ Méthode C: dans les balises

- Les feuilles de style permettent :
  - D'obtenir une présentation **homogène** sur tout un site en faisant appel sur toutes les pages à **une même définition de style**
  - De permettre le changement de l'aspect d'un site complet entier par **la seule modification de quelques lignes**
  - Une **plus grande lisibilité** du HTML, car les styles sont définis à part
  - Des **chargements de page plus rapides**, pour les mêmes raisons que précédemment
  - Un **positionnement plus rigoureux des éléments**
- Regrouper dans un même document des caractéristiques de mise en forme associées à des groupes d'éléments
- Définition de **plusieurs styles** possibles et héritage des styles en cascade
- Fournir une plus grande richesse d'éléments de style graphique afin d'améliorer l'apparence des documents HTML.

## 6 Définition d'un style

- Ensemble de règles en texte permettant de décrire l'aspect des éléments de la page
- Une règle CSS contient 2 éléments principaux :
  - Un **sélecteur** de style précisant à quelles balises appliquer le style
  - Une **déclaration** de style définie entre accolade "{}"
    - Une ou plusieurs **propriétés** : suivie du caractère ":"
    - Une ou plusieurs **valeurs** : séparée par des virgules et suivie du caractère ";"



Dans cet exemple, tous les éléments `<p>` seront alignés au centre, avec une couleur de texte rouge :

```
p {
```

```
    color: red;
    text-align: center;
}
```

Cet exemple de feuille de style:

- à fond blanc,
- avec une police de caractère par défaut Verdana, couleur noire,
- **un titre de premier niveau** bleu marine centré
- **un titre de deuxième niveau** rouge décalé de 15 pixels
- **des liens** passant du vert au gris avec un petit effet rouge non souligné au passage de la souris.

```
html, body {
    margin: 0;
    padding: 0;
}
body {
    background-color: white;
    font-family: Verdana, sans-serif;
    font-size: 100%;
}
h1 {
    font-size: 200%;
    color: navy;
    text-align: center;
}
h2 {
    font-size: 150%;
    color: red;
    padding-left: 15px;
}
p, ul, li, td {
    color: black;
}
a:link {
    color: green;
    text-decoration: underline;
}
a:visited {
    color: gray;
}
a:hover {
    color: red;
    text-decoration: none;
}
a:active, a:focus {
    color: red;
}
```

## 8 *Notion de cascades*

Les styles peuvent être déclarés à différents endroits, et selon ces endroits ils seront plus ou moins prioritaires.

➤ On obtient donc une cascade de styles.



Déclaration des styles dans une feuille de style **externe** : elle a le moins de poids

Déclaration des styles en **interne**, dans **l'en-tête** de la page . Les styles déclarés auront plus de poids que ceux de la feuille de style externe et donc l'emporteront en cas de conflits.

Déclaration des styles en **attributs** des éléments html. Ces styles l'emporteront sur tous les autres.



## 9 Enregistrement d'une feuille de style

- Il existe donc trois façons d'enregistrer les styles :
  1. dans une feuille de style externe
  2. dans une feuille de style interne
  3. dans le code au sein des balises html  
(déconseillé)

- Enregistrer le code CSS dans un fichier s'appelant (par exemple) "style.css", et mettre dans l'en-tête de la page html (entre les balises <head></head>) :

```
<link href="style.css" rel="stylesheet" media="all" type="text/css">
```

Pour lier le CSS à un fichier HTML, nous utilisons la balise **<link>** à l'intérieur de la section HTML <head> .

- l'attribut **type** est inutile avec un doctype HTML5
  - media="all" ➔ feuille sera lisible par tous les types de médias (écrans, TV, imprimante...).
  - media="screen" (écran),
  - media="print" (imprimante).
  - L'attribut **REL** est utilisé pour définir la relation entre le fichier lié et le document HTML.  
REL=StyleSheet **spécifie un style persistant** ou préféré tandis que REL="Alternate StyleSheet" définit un style alternatif.

- Pour déclarer des styles qui ne s'appliqueront qu'à la page considérée, les styles sont à déclarer entre les balises suivantes : `<style type="text/css" media="screen"> .... </style>`

Tout ceci est à placer, comme précédemment, entre les balises `<head></head>`.

```
<head>
<style type="text/css" media="screen">
    BODY {Background-color: pink;}

    p {
        color: blue;
        text-align: center;
    }
</style>
</head>
```

Les feuilles de style permettent : D'obtenir une présentation homogène sur tout un site en faisant appel sur toutes les pages à une même définition de style De permettre le changement de l'aspect complet entier par la seule modification de quelques lignes Une plus grande lisibilité du HTML, car les styles sont définis à part Des chargements de page plus rapides, pour les mêmes raisons que précédemment Un positionnement plus rigoureux des éléments Regrouper dans un même document des caractéristiques de mise en forme associées à des groupes d'éléments Définition de plusieurs styles possibles des styles en cascade Fournir une plus grande richesse d'éléments de style graphique afin d'améliorer l'apparence des documents HTML.

## 12 Styles CSS dans le code html ("inline")

- De façon encore plus ponctuelle, si l'on veut attribuer un style à un seul endroit, on peut **déclarer le style à l'intérieur** d'une balise html à l'aide de l'**attribut style**.
- Par exemple :

```
<p style="text-align:center; color:red">  
  Bla bla bla...  
</p>
```

### METTRE EN FORME UN TEXTE EN CSS

- Attributs utilisés :
  - font ; font-family ; font-weight ; font-variant ; font-style: (Police de caractère)
  - color
  - text-align ; text-decoration ; text-transform ( style de texte)
  - line-height (hauteur de la ligne)

## Choix des polices de caractères

- Il existe 5 familles de polices de caractères dites "génériques" :
  - Serif
  - Sans-serif
  - Monospace
  - Cursive
  - Fantasy

**Arial Black**  
**Verdana Impact**  
Trebuchet MS  
Helvetica Geneva

POLICES SANS-SERIF

Courier  
Courier New  
Monaco  
Lucida

MONOSPACE

Palatino  
Georgia Times  
Garamond  
**COPPERPLATE**

POLICES SERIF

Comic Sans MS  
Zapf Chancery  
Brush Script

CURSIVE (MANUSCRIT)

## Famille Serif

- Les fontes de caractères de la famille serif assez courantes sont :
  - Times new roman (PC)
  - Times (Mac)
  - Georgia (Mac/PC)
  - Palatino Linotype (PC)
  - Palatino (Mac)
- Elles n'ont pas toutes le même aspect, les associations à respecter pour une homogénéité de rendu pourraient donc être :

```
body { font-family:"times new roman", times, serif; }
```

```
body { font-family:georgia, serif; }
```

```
body { font-family:"palatino linotype", palatino, serif; }
```

## Famille Sans-serif

- Verdana (Mac/PC)
- Arial (Mac/PC)
- Trebuchet (PC)
- Helvetica (Mac)
- Tahoma (PC)
- Geneva (Mac)
- Pour ces fontes sans-serif, les associations pourraient être :

```
body { font-family:arial,sans-serif; }
```

```
body { font-family:verdana,sans-serif; }
```

```
body { font-family:trebuchet, helvetica, sans-serif; }
```

```
body { font-family:tahoma, geneva, sans-serif; }
```

## Unités de taille de caractères

- Ce qu'il ne faut pas utiliser :
  - On n'utilise pas de tailles de caractères fixes comme les points (**pt**), les picas (**pc**) ou les centimètres (**cm**).
    - ➔ Ils ne sont pas redimensionnables à l'écran, or on doit toujours laisser la possibilité au visiteur de zoomer la page.
  - On évite aussi d'utiliser les pixels (**px**) car Internet Explorer les considère comme une unité de taille de caractère fixe, on se retrouve donc avec le même problème que précédemment.
- Il faut donc utiliser des unités relatives, telles que les **em** ou les **%**. Ces unités sont proportionnelles à la taille en pixels déclarée dans le navigateur. Par défaut, ceux-ci sont en général réglés à 16px.
- **La taille des caractères se déclare par l'attribut font-size.**

**font-size: 12px;**

**font-size: 0.8em; Valeurs proportionnelles**

**font-size: 80%; Type <percentage>**



## L'héritage

- Exemple

```
body { font-family:arial, sans-serif; font-size:90%; } p { font-size:90%; }
```

➔ Les textes inclus dans les paragraphes n'auront pas une taille de 90% des 16 pixels déclarés par défaut (soit 14,4 pixels), mais 90% des 90% des 16 pixels (soit à peu près 13 pixels...

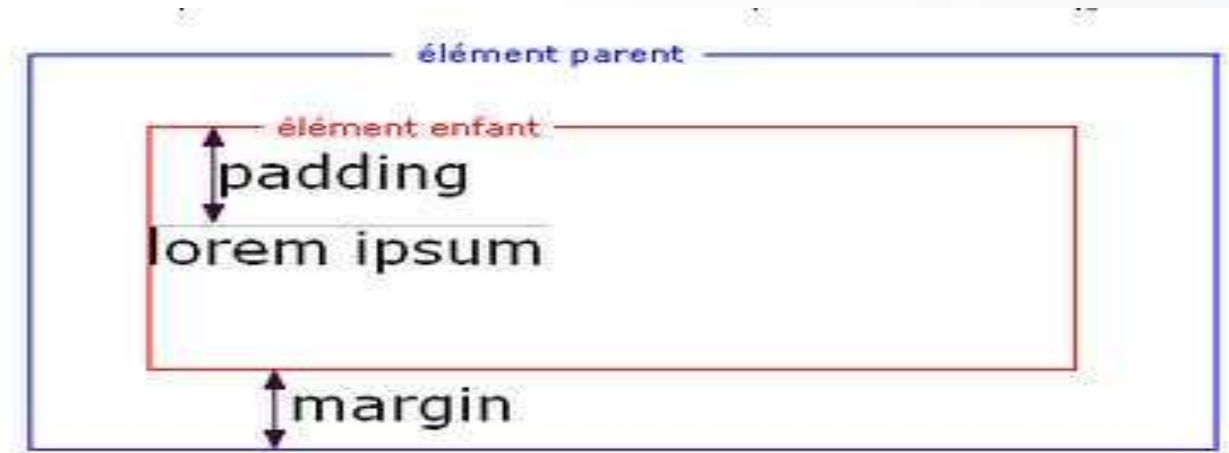
En effet, le paragraphe hérite des propriétés de son/ses parent(s).

Solution

```
body { font-family:arial, sans-serif; font-size:100%; }  
h1 { font-size:200%; }  
#footer p { font-size:90%; }
```

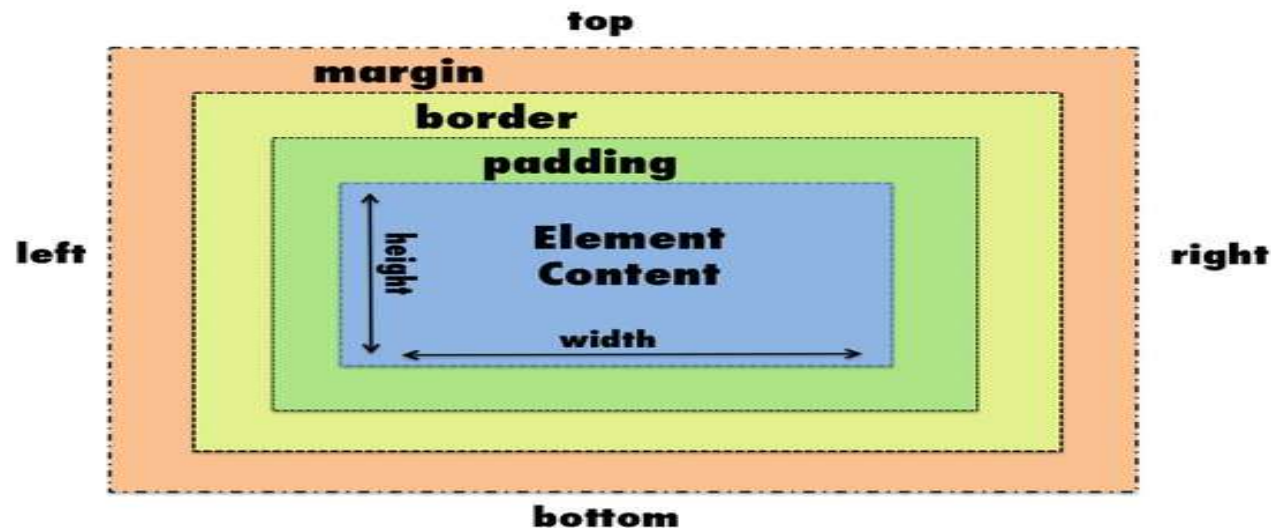
## GÉRER LES MARGES EN CSS

- Propriétés utilisées :
  - margin
  - Padding



## Définitions des marges intérieures et extérieures

Pour chaque élément html on peut donc définir **l'espacement** qui le séparera des autres éléments (margin) et les espacements intérieurs dont il peut bénéficier (padding).



## 19 Feuille de style

- ❖ Les margin propriétés CSS sont utilisées pour créer un espace autour des éléments, en dehors de toute bordure définie. Il existe des propriétés permettant de définir la marge de chaque côté d'un élément (haut, droite, bas et gauche).

Description	propriété
<b>margin</b>	Une propriété raccourcie pour définir toutes les propriétés de marge dans une seule déclaration
<b>margin-bottom</b>	Définit la marge inférieure d'un élément
<b>margin-left</b>	Définit la marge gauche d'un élément
<b>margin-right</b>	Définit la marge droite d'un élément
<b>margin-top</b>	Définit la marge supérieure d'un élément



```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
div {
```

```
border: 1px solid black;
```

```
margin-top: 200px;
```

```
margin-bottom: 100px;
```

```
margin-right: 330px;
```

```
margin-left: 80px;
```

```
background-color: lightblue;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h2>Using individual margin properties</h2>
```

```
<div>This div element has a top margin of 200px, a right margin  
of 330px, a bottom margin of 100px, and a left margin of  
80px.</div>
```

```
</body>
```

```
</html>
```

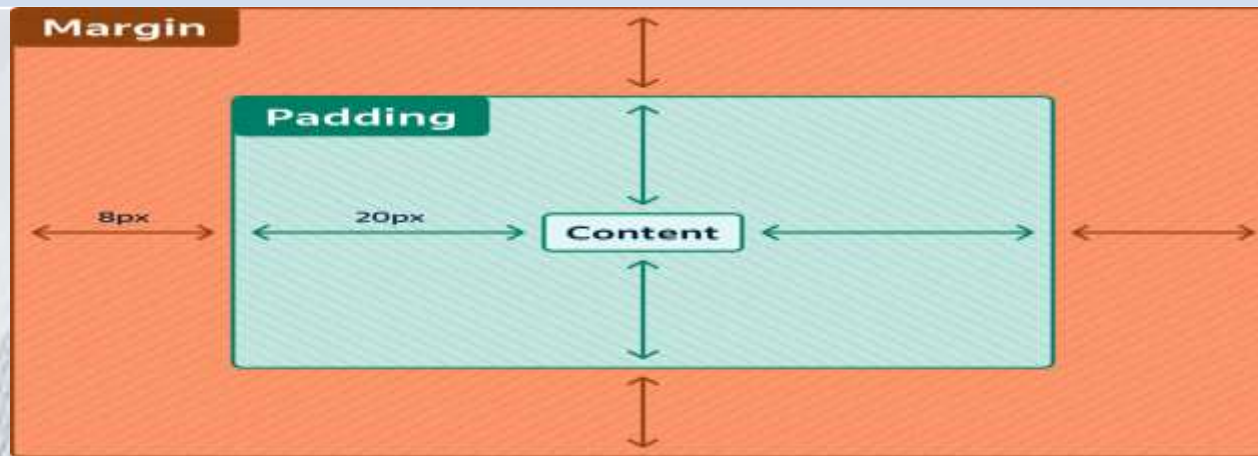
## Using individual margin properties

This div element has a top margin of 100px, a right margin of 150px, a bottom margin of 100px, and a left margin of 80px.

## 20 Feuille de style

- Les **padding** propriétés CSS sont utilisées pour générer un espace autour du contenu d'un élément, à l'intérieur de toute bordure définie.

Description	propriété
<b>padding</b>	Une propriété raccourcie pour définir toutes les propriétés de remplissage dans une seule déclaration
<b>padding-bottom</b>	Définit le remplissage inférieure d'un élément
<b>padding-left</b>	Définit le remplissage gauche d'un élément
<b>padding-right</b>	Définit le remplissage droite d'un élément
<b>padding-top</b>	Définit le remplissage supérieure d'un élément



```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  background-color: lightblue;
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 100px;
  padding-left: 80px;
}
```

```
</style>
</head>
<body>

<h2>Using individual padding properties</h2>

<div>This div element has a top padding of 50px, a right
padding of 30px, a bottom padding of 50px, and a left padding
of 80px.</div>

</body>
</html>
```

## Using individual padding properties

This div element has a top padding of 50px, a right padding of 30px, a bottom padding of 50px, and a left padding of 80px.

## Déclaration des tailles

- Les tailles de ces marges peuvent se déclarer en pixels (**px**), en **em**, en **%**, etc.
- On peut détailler les tailles des marges à l'aide des suffixes -top (haut), -right (droite), -bottom(bas), -left (gauche), ou synthétiser les quatre d'un seul coup (la première valeur étant celle du haut, puis on tourne dans le sens des aiguilles d'une montre).

`margin:2px 5px 2em 0;` revient à :

`margin-top:2px; margin-right:5px; margin-bottom:2em; margin-left:0;`

Si on ne met que deux valeurs, la 1<sup>ère</sup> s'appliquera au haut et au bas, la seconde à droite et à gauche.

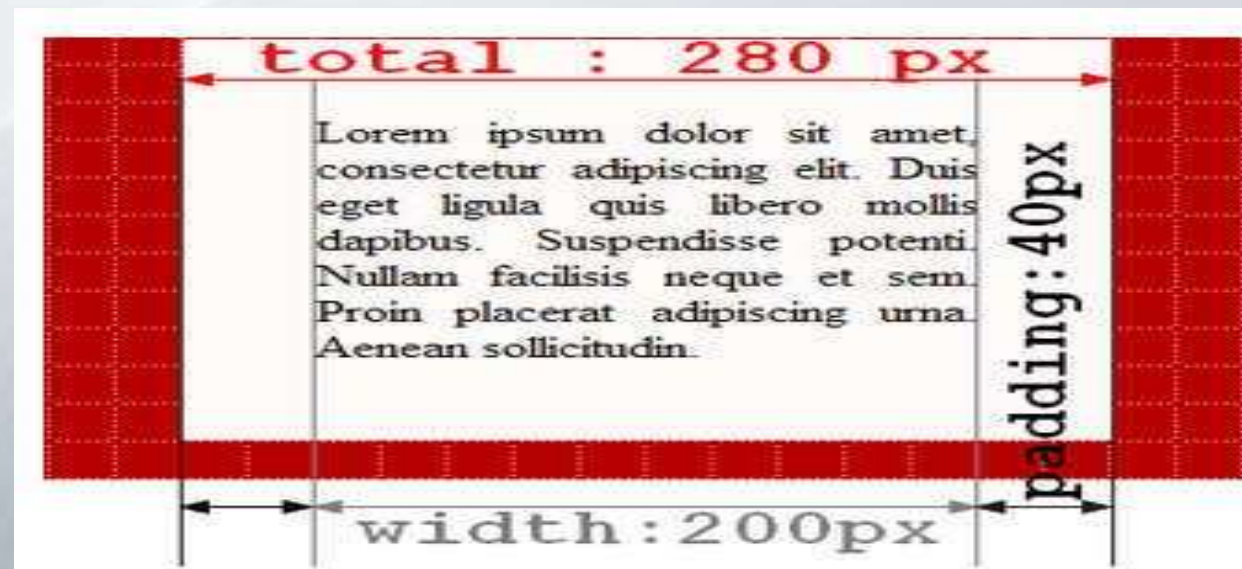
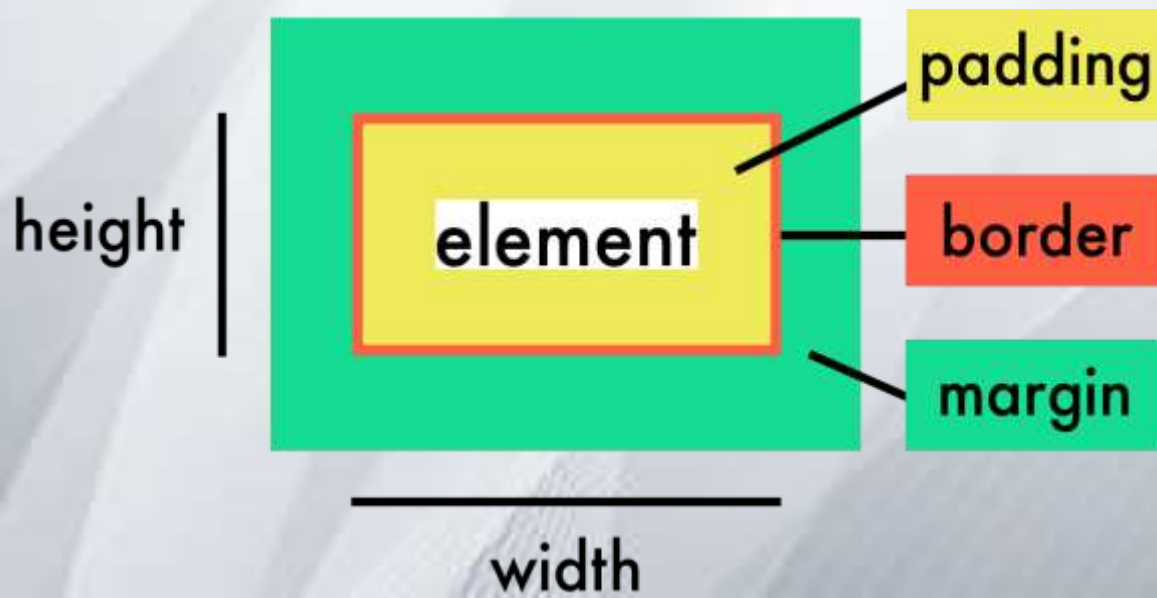
`padding:2px 5px;` revient à :

`padding-top:2px; padding-bottom:2px; padding-right:5px padding-left:5px;`



- D'après le "**box model**", lorsqu'on attribue une taille à un élément de type block (de width ou height), les marges viennent s'ajouter à cette taille.
- Notez aussi le très pratique **margin:auto** qui permet, dans le cas d'un bloc muni d'une largeur, de centrer horizontalement un élément.
- Exemple 

```
p { width:200px; padding:40px; margin:auto; }
```
- Le paragraphe aura une largeur totale de  $200 + 2 \times 40 = 280$  pixels et sera centré dans son élément parent.





- Propriétés utilisés : background-color , float , width , margin , text-align

## 1 - SÉLECTEURS CLASS

- dans une feuille de style, le nom du sélecteur class est toujours **précédé d'un point.**
- Exemple

- Code CSS

```
.haut {text-align:right;}
```

- Code HTML

Pour appeler ce style dans la page html, on indique simplement class="haut" à l'intérieur de la balise voulue.

```
<p class="haut"> <a href="#haut">Haut de page</a> </p>
```

## 24 SÉLECTEUR CLASS

```
<!DOCTYPE html>
<html>
<head>
<style>
.red {
  color: #f33;
}

.yellow-bg {
  background: #ffa;
}

.fancy {
  font-weight: bold;
  text-shadow: 4px 4px 3px #77f;
}
</style>
</head>

<body>
  <p class="red">This paragraph has red text.
</p>
  <p class="red yellow-bg">
    This paragraph has red text and a yellow
    background.
  </p>
  <p class="red fancy">This paragraph has red
  text and "fancy" styling.</p>
  <p>This is just a regular paragraph.</p>
</body>
</html>
```

This paragraph has red text.

This paragraph has red text and a yellow background.

This paragraph has red text and "fancy" styling.

This is just a regular paragraph.

# 25 SÉLECTEURS CSS CLASS ET ID

## 2 – SÉLECTEUR id

- Le sélecteur id a presque la même fonction, à la différence importante qu'on ne peut **l'utiliser qu'une seule fois dans la page**, contrairement au sélecteur class. C'est pour cela qu'il est plutôt utilisé à la mise en page qu'à la mise en forme de caractères
- Exemple
  - Code CSS
  - Code (X)HTML

```
#menu { background-color:silver; width:100px; float:left; } #contenu { margin-left:110px; }
```

```
<h1 id="haut">Exemple des sélecteurs "class" et "id"</h1>
```

```
<div id="menu">
```

```
<ul> <li>item 1</li> <li>item 2</li> <li>item 3</li> <li>item 4</li> <li>item 5</li>  
</ul> </div>
```

```
<div id="contenu">
```

```
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam  
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat
```

```
</p>  
</div>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
#identified {
```

```
  background-color: skyblue;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
  <div id="identified">This div has a special ID on it!</div>
```

```
  <div>This is just a regular div.</div>
```

```
</body>
```

```
</html>
```

This div has a special ID on it!

This is just a regular div.

## 26 MISES EN PAGE EN COLONNES SANS TABLEAU

Le principe est très simple :

- **diviser** chaque élément de la page par un **div**.
- La page se divise en **quatre sections**, chacune correspondant à une zone précise de la page :

Un en-tête : `div id="entete"`

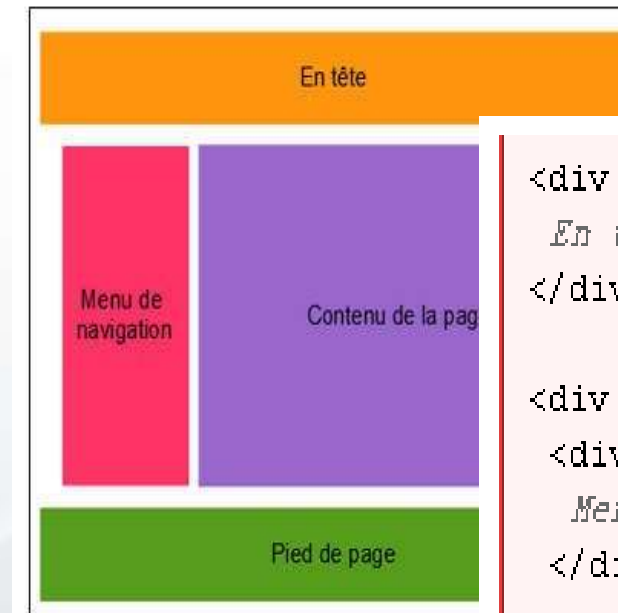
Un menu : `div id="menu"`

Le contenu : `div id="contenu"`

Un pied de page : `div id="footer"`

- Les boîtes menu et contenu seront englobées dans un autre **cadre main**.

➤ Le but est de placer les cadres menu et contenu l'un à côté de l'autre



```
<div id="entete">  
  En tête  
</div>
```

```
<div id="main">  
  <div id="menu">  
    Menu  
  </div>
```

```
<div id="contenu">  
  Contenu  
</div>  
</div>
```

```
<div id="footer">  
  Pied de Page  
</div>
```

# 27 Enpositionnement flottant

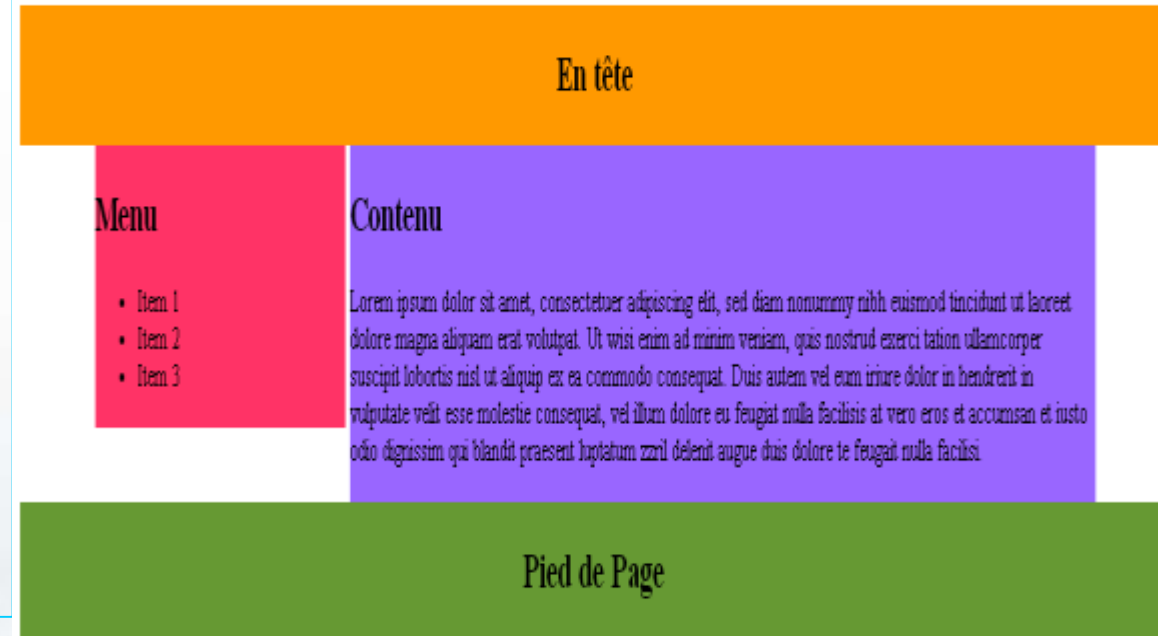
- Attributs utilisés : **clear** , **float** , **margin** , **max-width** ; **width** , **Padding** , **text-align**

- Code CSS

```
#entete, #menu, #contenu, #footer { padding:1px 0; }  
#entete { background-color:#FF9900; text-align:center;}  
#main { max-with:960px; margin:auto; }  
#menu { float:left; width:240px; background-color:#FF3366; }  
#contenu { margin-left:245px; background-color:#9966FF; }  
#footer { background-color:#669933; text-align:center; clear:both; }
```

- On peut aussi positionner le **menu à droite** sans toucher au code html ! Ce qui est bien sûr impossible avec une mise en page par tableaux.

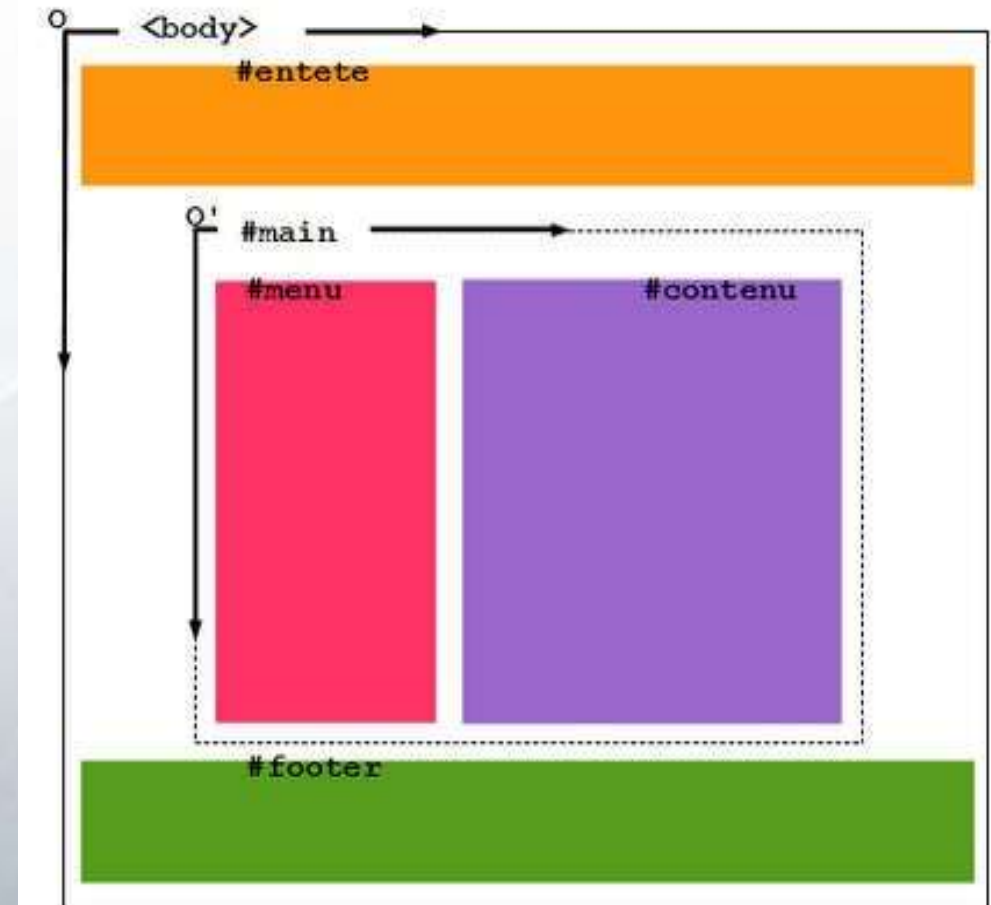
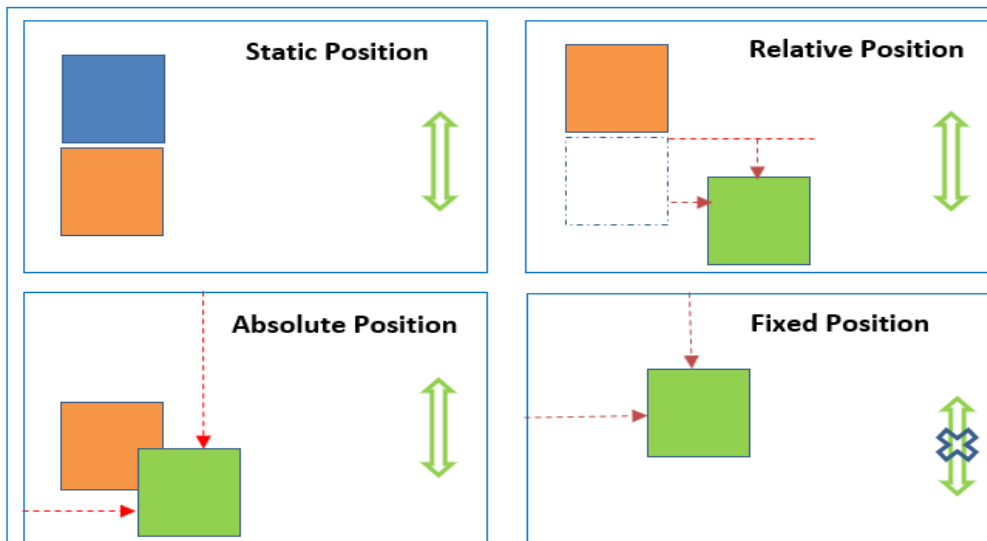
```
#menu { float:right;  
width:240px; }  
#contenu { margin-  
right:245px; }
```



## 28 En positionnement absolu

Attributs utilisés : min-height , position

- L'origine du repère à partir duquel on peut **positionner l'élément**, à l'aide des propriétés **top**, **bottom**, **left** ou **right**, se fait à partir du dernier ancêtre positionné, ou, à défaut, **du coin haut à gauche** de la fenêtre.
  - Par défaut **l'origine O est en haut à gauche**.
  - Mais on peut effectuer **un changement de repère en O'** à l'aide d'un **élément parent aux éléments à positionner**.
- Ici le main (bloc parent en position:relative) va permettre de changer de repère pour menu et/ou contenu (éléments enfants).



```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
.box {
```

```
  display: inline-block;
```

```
  width: 100px;
```

```
  height: 100px;
```

```
  background: red;
```

```
  color: white;
```

```
}
```

```
#deux {
```

```
  position: relative;
```

```
  top: 20px;
```

```
  left: 20px;
```

```
  background: blue;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
  <div class="box" id="un">Un</div>
```

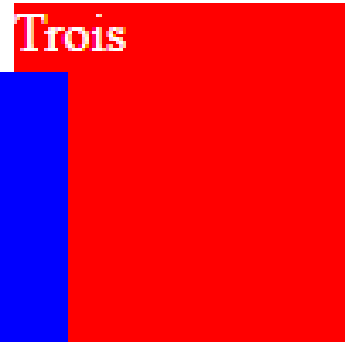
```
  <div class="box" id="deux">Deux</div>
```

```
  <div class="box" id="trois">Trois</div>
```

```
  <div class="box" id="quatre">Quatre</div>
```

```
</body>
```

```
</html>
```





**BORDURES EN CSS** border-width , border-style , border-color , padding

**IMAGE DE FOND EN CSS** background-image , background-repeat , background-position

**EFFETS ROLLOVER À L'AIDE DE :HOVER** Color, text-decoration , background-color , background-image

Le :hover sélecteur permet de sélectionner des éléments lorsque vous passez la souris sur eux.

- Exemples

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

```
a:hover { color:red; text-decoration:none;
```

```
background-color:black; }
```

```
a:hover { color:red; text-decoration:none;
```

```
background-image:url(images/gris_anim.gif);
```

```
background-repeat:no-repeat; }
```

### Cumuler image de fond et caractère

```
#menu a:hover { background-  
    image:url(images/gris_anim.gif); background-  
    repeat:no-repeat; }
```

```
#menu a:hover:before { content:" » "; }
```

- Première technique : version classique

- Attributs utilisés :
  - color
  - display
  - font-family ; font-size
  - list-style-image ; list-style-position

- **Exemple**

```
li { font-family: Arial, sans-serif;  
font-size: 100%; color: black;  
display : list-item; list-style-  
image : url(puce.gif);  
}
```

- Deuxième technique : variante Made-in Sam

- Attributs utilisés :
  - background-image ; background-repeat ; background-position
  - color
  - display
  - font-family ; font-size
  - list-style-type
  - padding

- Exemple

```
li { font-family: Arial, sans-serif;  
font-size: 100%; color: black; list-  
style-type: none; background-  
image: url(puce.gif);  
background-repeat: no-repeat;  
background-position: 0 0.32em;  
padding-left: 15px; }
```

# Nouvelles spécifications de CSS3

- CSS3 contient les "anciennes spécifications CSS2".
- De plus, de nouveaux modules sont ajoutés.
- Certains des modules CSS3 les plus importants sont:
  - Sélecteurs
  - Modèle de boîte
  - Fonds et bordures
  - Effets de texte
  - Transformations 2D / 3D
  - Animations
  - Disposition des colonnes multiples
  - ...
- La plupart des nouvelles propriétés CSS3 sont implémentées dans des navigateurs modernes.

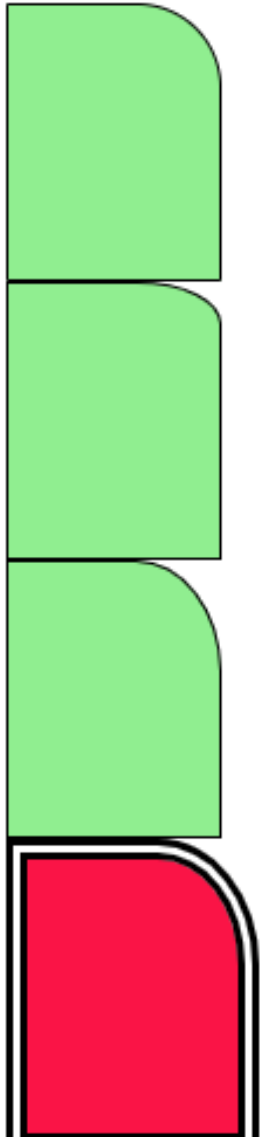


- Avec la nouvelle propriété **border-radius** de CSS3, on peut avoir des éléments avec des coins arrondis .

```
#corner {  
  border-radius: 25px;  
  border: 2px solid #73AD21;  
  padding: 20px;  
  width: 200px;  
  height: 150px;  
}
```

Rounded corner

```
.arc_cercle {  
  border-top-right-radius: 40px 40px;  
}  
.arc_ellipse {  
  border-top-right-radius: 40px 20px;  
}  
.pourcentage {  
  border-top-right-radius: 40%;  
}  
.rognage {  
  border: black 10px double;  
  border-top-right-radius: 40%;  
  background-color: rgb(250,20,70);  
  background-clip: content-box;  
}
```



- Avec la nouvelle propriété **border-image** de CSS3, on peut définir une image à utiliser comme bordure autour d'un élément.
- La propriété comporte trois parties:
  1. L'image à utiliser comme bordure
  2. Où découper l'image
  3. Définir si les sections du milieu doivent être répétées ou étirées

Propriété	Description
<u><a href="#">border-image</a></u>	Une propriété abrégée pour mettre toutes les valeurs des autres propriétés (ci-dessous)
<u><a href="#">border-image-source</a></u>	Spécifie L'url de l'image de bordure
<u><a href="#">border-image-slice</a></u>	Spécifie comment découper l'image de bordure
<u><a href="#">border-image-width</a></u>	Spécifie les largeurs of the l'image de bordure
<u><a href="#">border-image-repeat</a></u>	Spécifie si l'image de bordure sera répétée, arrondie ou étirée

## 34 css3 : Bordure image

- Avec la nouvelle propriété **border-image** de CSS3, on peut définir une image à utiliser comme bordure autour d'un élément.
- La propriété comporte trois parties:
  1. L'image à utiliser comme bordure
  2. Où découper l'image
  3. Définir si les sections du milieu doivent être répétées ou étirées
- Soit l'exemple suivant :
  - Supposant on a l'image suivante (border.png) à utiliser pour la bordure :
- La propriété border-image prend l'image et la découpe en 9 sections
- Elle place alors les coins aux coins et les sections du milieu sont répétées ou étirées comme vous spécifier .

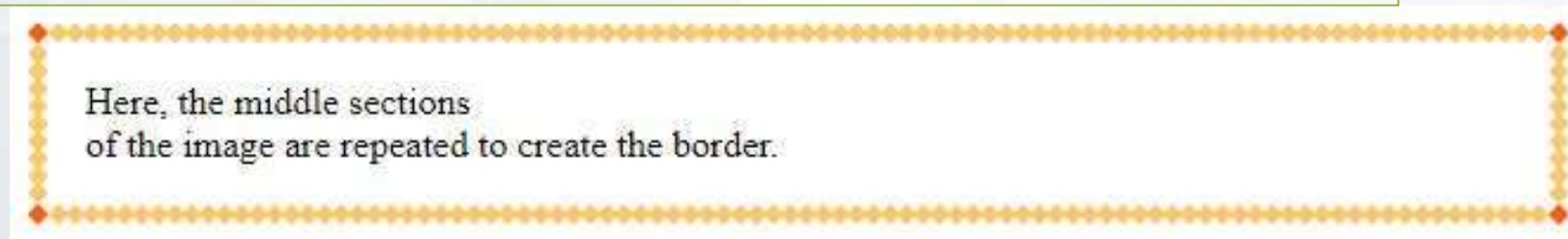


➔ Remarque: Pour que l'image de bordure fonctionne, l'élément a également besoin de la propriété de bordure définie!

## 35 CSS3: Bordure image

- Ici, les sections intermédiaires de l'image sont répétées pour créer la bordure:

```
#borderimg {  
  border: 10px  
  solidtransparent;  
  padding: 15px;  
  border-image: url(border.png) 30round;  
}
```



```
#borderimg {  
  border: 50px solid transparent;  
  padding: 15px;  
  border-image: url(border.png) 36 round;  
}
```





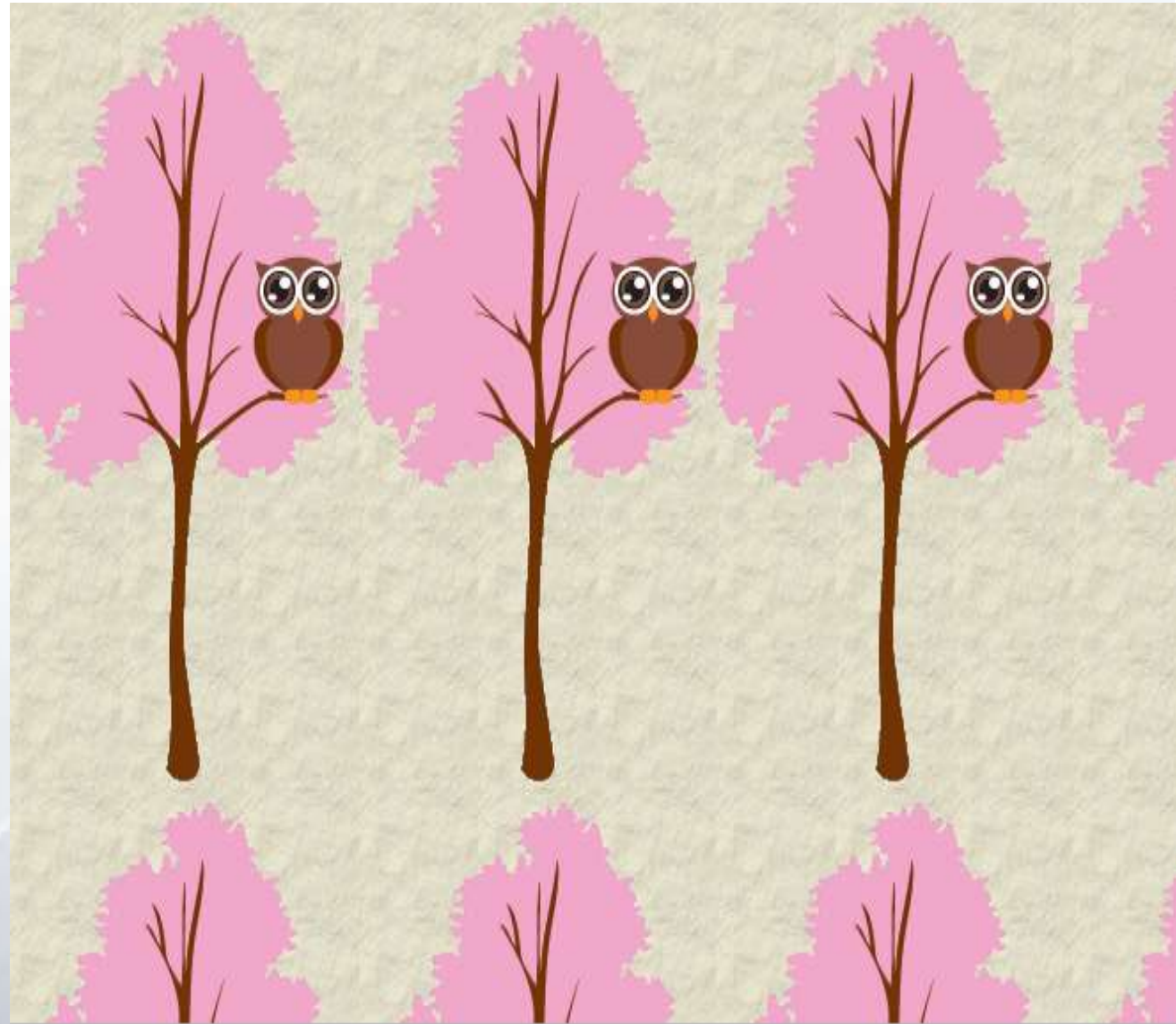
- CSS3 offre quelques nouvelles propriétés d'arrière-plan (***background***), qui permettent une plus grande commande de l'élément *background*.
- Avec CSS3 **on peut définir plusieurs images d'arrière-plan à un élément.**
- Les nouvelles propriétés CSS3:
  1. *background-size*
  2. *background-origin*
  3. *background-clip*

### 1-Arrière-plans : plusieurs images

- CSS3 permet d'ajouter plusieurs images d'arrière-plan pour un élément, à travers la propriété ***background-image***.
- Les différentes images d'arrière-plan sont séparées par **des virgules**, et les images sont empilées les **unes au dessous des autres**.

## 38 Arrière-plans

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("img_tree.gif")
, url("paper.gif") ;
  background-color: #cccccc;
}
</style>
</head>
<body>
</body>
</html>
```



- La propriété CSS3 **background-size** permet de spécifier la taille des images d'arrière-plan.
- Avant CSS3, la taille d'une image d'arrière-plan était la taille réelle de l'image.
- CSS3 permet de réutiliser les images d'arrière-plan dans différents contextes.
- La taille peut être spécifiée en longueurs, en pourcentage ou en utilisant l'un des deux mots-clés: **contenir** ou **couvrir**.
- L'exemple suivant **redimensionne** une image d'arrière-plan à beaucoup plus petite que l'image originale (à l'aide de pixels):

## Lorem Ipsum Dolor

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis commodo consequat.

```
#div1 {  
  background: url(img_flower.jpg);  
  background-size: 100px 80px;  
  background-repeat: no-repeat;  
}
```

## Lorem Ipsum Dolor

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

- La valeur **cover** indique que l'image d'arrière-plan devrait être aussi petite que possible tout en ayant ses deux dimensions supérieures ou égales à celles du conteneur. Vous pouvez redimensionner l'exemple qui suit pour observer l'effet obtenu.
- La valeur ***contain*** indique que, quelle que soit la taille de la boîte englobante, l'image d'arrière-plan devrait être redimensionnée afin que chaque côté soit aussi grand que possible, sans dépasser la taille du conteneur. Redimensionnez l'exemple qui suit pour observer l'effet obtenu.

```
<!DOCTYPE html>
<html>
<head>
<style>
#example1 {
  border: 2px solid black;
  padding: 25px;
  background: url(mountain.jpg);
  background-repeat: no-repeat;

  background-size: contain;
}

#example2 {
  border: 2px solid black;
  padding: 25px;
  background: url(mountain.jpg);
  background-repeat: no-repeat;
  background-size: cover;
}
</style>
</head>
<body>
```

```
<h2>background-size: contain</h2>
<div id="example1">
  <h2>Hello World</h2>
  <p>The background image ...</p>
</div>

<h2>background-size:cover:</h2>
<div id="example2">
  <h2>Hello World</h2>
  <p>Here, the background image ...</p>
</div>

</body>
</html>
```



background-size: contain



background-size:cover:





- la propriété **background-origin** de CSS3 spécifie où l'image d'arrière-plan est positionnée.
- La propriété prend trois valeurs différentes:
  - padding-box** : (par défaut) l'image d'arrière-plan commence à partir **du coin supérieur gauche du bord interne** (padding)
  - border-box** : l'image d'arrière-plan commence à partir **du coin supérieur gauche de la bordure**
  - content-box** : l'image d'arrière-plan commence à partir du coin **supérieur gauche du contenu**

No background-origin (padding-box is default):



background-origin: border-box:



background-origin: content-box:



- CSS prend en charge les noms de couleurs, les couleurs hexadécimales et RVB.
- En outre, CSS3 présente également:



### 1- Couleurs RGBA

- Les valeurs de couleur RGBA sont une extension des valeurs de couleur RGB avec une chaîne alpha, qui spécifie l'opacité d'une couleur.
- Une valeur de couleur RGBA est spécifiée avec: rgba (rouge, vert, bleu, alpha). Le paramètre alpha est un nombre entre 0.0 (entièrement transparent) et 1.0 (entièrement opaque).

```
rgba(255, 0, 0, 0.2);
```

```
rgba(255, 0, 0, 0.4);
```

```
rgba(255, 0, 0, 0.6);
```

```
rgba(255, 0, 0, 0.8);
```

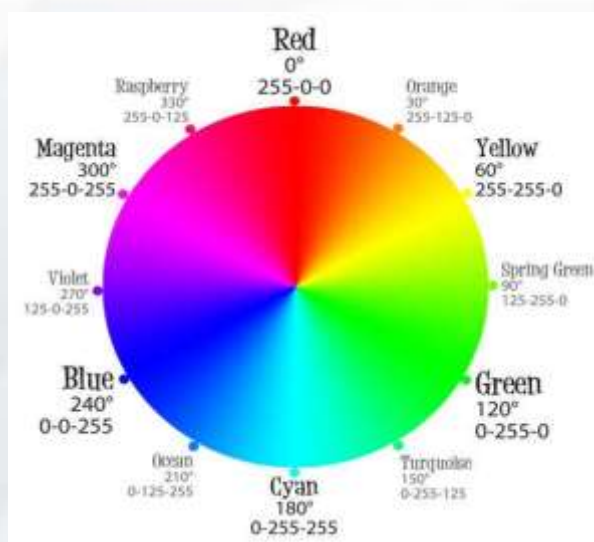
```
#p1 {background-color: rgba(255, 0, 0, 0.3);} /* red with opacity */  
#p2 {background-color: rgba(0, 255, 0, 0.3);} /* green with opacity */  
#p3 {background-color: rgba(0, 0, 255, 0.3);} /* blue with opacity */
```

# 45 Les couleurs

## 2- Couleurs TSL (HSL)

- HSL signifie teinte (Hue), Saturation (Saturation) et Luminance (Lightness).
- Une valeur de couleur HSL est spécifiée avec: hsl (teinte, saturation, Luminance).
- La **teinte** est un degré sur la roue de couleur (de 0 à 360):

- 0 (ou 360) est rouge
- 120 est vert
- 240 est bleu



- La **saturation** est une valeur de pourcentage: 100% est la pleine couleur.
- La **luminance** est également un pourcentage; 0% est sombre (noir) et 100% blanc

## Exemple

```
<style>
#p1    {background-color:hsl(120,100%,50%);}
#p2    {background-color:hsl(120,100%,75%);}
#p3    {background-color:hsl(120,100%,25%);}
#p5    {background-color:hsl(290,100%,50%);}
#p6 {background-color:hsl(290,60%,70%);}
</style>
```

```
<body>
<p>HSL colors:</p>
<p id="p1">Green</p>
<p id="p2">Light green</p>
<p id="p3">Dark green</p>
<p id="p5">Violet</p>
<p id="p6">Pastel violet</p>
</body>
```

HSL colors:

Green

Light green

Dark green

Violet

Pastel violet

# 46 Les couleurs

## 3- Couleurs HSLA

- HSLA est une extension HSL avec une chaîne **alpha**, qui spécifie l'opacité d'une couleur.
- Une valeur de couleur HSLA est spécifiée avec:
  - **hsla (teinte, saturation, luminance, alpha)**, où le paramètre alpha définit l'opacité. Le paramètre alpha est un nombre entre 0.0 (entièrement transparent) et 1.0 (entièrement opaque).

`hsla(0, 100%, 30%, 0.3);`

`hsla(0, 100%, 50%, 0.3);`

`hsla(0, 100%, 70%, 0.3);`

`hsla(0, 100%, 90%, 0.3);`

## 4- l'opacité

```
div {  
  background-color: yellow;  
}  
.leger {  
  /* On ne voit presque pas le texte */  
  opacity: 0.2;  
}  
.moyen {  
  /* On peut mieux discerner le texte */  
  opacity: 0.5;  
}  
.lourd {  
  /* Le texte est clairement visible */  
  opacity: 0.9;  
}
```

On arrive à peine à lire.

On voit mieux.

Ceci est plus simple à lire.



- Les couleurs dégradées (gradients) CSS3 permettent d'afficher des transitions progressives entre deux ou plusieurs couleurs spécifiées.
    - Avant css3, on devait utiliser des images pour ces effets.
  - Cependant, en utilisant des gradients CSS3, on peut réduire le temps de téléchargement et l'utilisation de la bande passante.
  - Les éléments avec des gradients sont plus nets lorsqu'ils sont agrandis
    - car le dégradé est généré par le navigateur.
- CSS3 définit deux types de gradients:
    1. Gradients linéaires (haut vers le bas (par défaut) / gauche / droite / en diagonale)
    2. Gradients radiaux (définis par leur centre)

## 1- Les gradients linéaires

- Pour créer un gradient linéaire, il doit définir **au moins deux arrêts de couleur**.
- Les arrêts de couleur sont les couleurs dont on souhaite effectuer des transitions progressives
- On peut aussi définir un point de départ et une direction (ou un angle) avec l'effet de dégradé.


```
background: linear-gradient(direction, color-stop1, color-stop2, ...);
```

```
#grad {  
  background: red; /* For browsers that do not support gradients */  
  background: linear-gradient(red, yellow); /* Standard syntax */  
}
```




- Les gradients linéaires : droite → gauche

```
#grad {  
  background: red; /* For browsers that do not support gradients */  
  background: linear-gradient(to right, red , yellow); /*Standard */  
}
```



```
#grad {  
  background: red; /* For browsers that do not support gradients */  
  background: linear-gradient(to right, red,orange,yellow,green,blue,indigo,violet);  
}
```



# Les couleurs dégradées

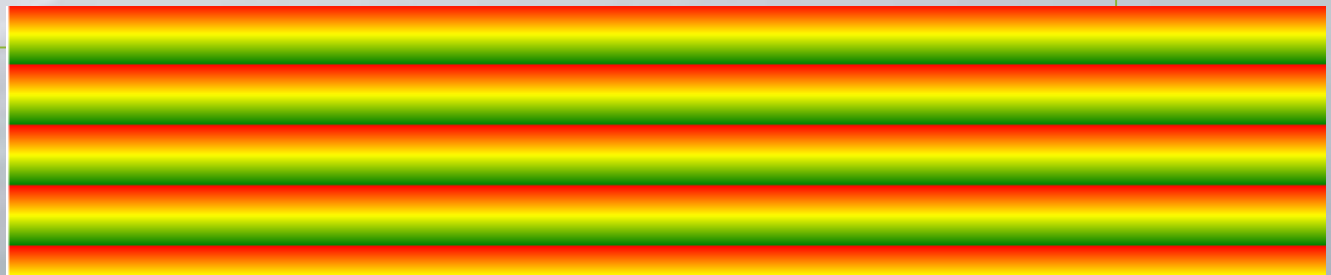
- Les gradients linéaires : diagonale

```
#grad {  
  background: red; /* For browsers that do not support gradients  
  
  */  
  background: linear-gradient(to bottom right, red, yellow); /* Standard syntax */  
}
```



- Répéter Les gradients linéaires

```
#grad {  
  background: red; /* For browsers that do not support gradients */  
  background: repeating-linear-gradient(red, yellow 10%, green 20%);  
}
```



- **Les gradients radiaux**

- Un gradient radial est défini **par son centre**.
- Pour créer un gradient radial, on doit également définir au moins **deux arrêts de couleur**.

```
background: radial-gradient(shape size at position, start-color, ..., last-color);
```

```
#grad {  
  background: red; /* For browsers that do not support gradients */  
  background: radial-gradient(red, yellow, green); /* Standard syntax */  
}
```

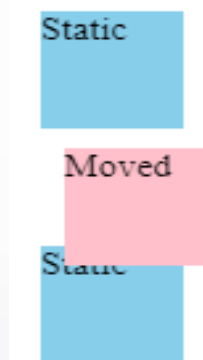


- Les transformations CSS3 permettent de:
  - **faire pivoter**
  - **étaler**
  - **d'éloigner les éléments.**
- Une transformation est un effet qui permet à un élément de changer de forme, de taille et de position.
- CSS3 prend en charge les transformations 2D et 3D.
- Pour les transformations 2D de css3 on utilise les méthodes suivantes :



## LesTransformations 2D : translation

- La méthode **translate ()** déplace un élément de sa position actuelle (selon les paramètres donnés pour l'axe X et l'axe Y).



```
div {
  transform: translate(50px, 100px);
}
```

```
div {
  width: 60px;
  height: 60px;
  background-color: skyblue;
}
.moved {
  transform: translate(10px, 10px);
  background-color: pink;
}
```

## LesTransformations 2D : rotation

- La méthode **rotate ()** fait pivoter un élément dans le sens des aiguilles d'une montre ou dans le sens inverse des aiguilles d'une montre selon un degré donné.
- L'exemple suivant fait pivoter l'élément <div> dans le sens des aiguilles d'une montre avec 20 degrés:



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

## LesTransformations 2D: taille

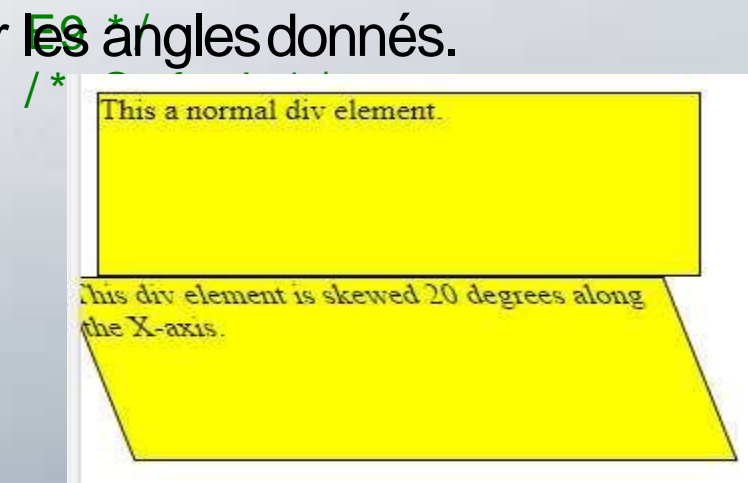
- La méthode **scale ()** augmente ou diminue la taille d'un élément (selon les paramètres donnés pour la largeur et la hauteur).
- L'exemple suivant augmente l'élément <div> pour être deux fois sa largeur d'origine et trois fois sa hauteur d'origine:

```
div {transform: scale(2, 3);}
```

## LesTransformations 2D: inclinaison

- La méthode **skewX ()/ skewY ()** permet de' incliner un élément selon l'axe des X/ des Y par l'angle donné.
- La méthode **skew()** déforme un élément le long de l'axe X et Y par les angles donnés.
- L'exemple suivant incline l'élément <div> 20 degrés selon l'axe X:

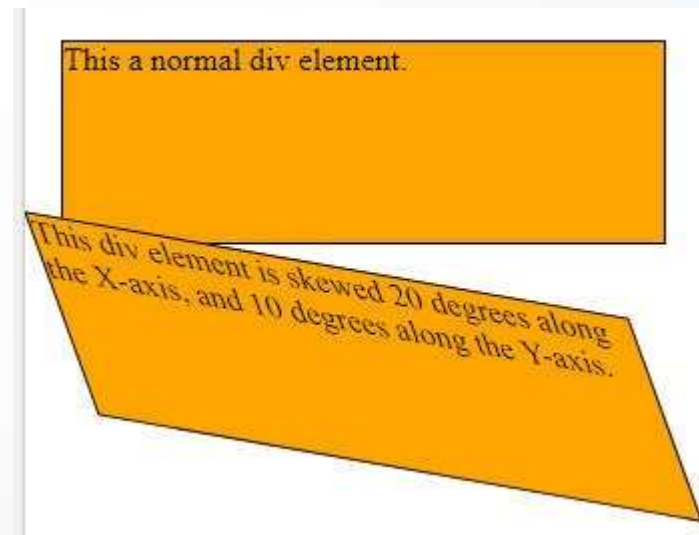
```
div {  
  transform: skewX(20deg);  
}
```





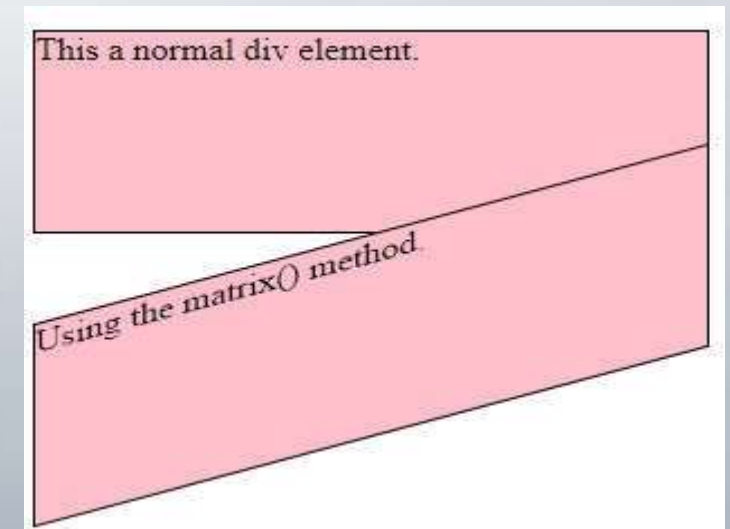
- La méthode **skew()** déforme un élément le long de l'axe X et Y par les angles donnés.

```
div {  
  transform: skew(20deg, 10deg);  
}
```



- La méthode **matrix()** combine toutes les méthodes de transformation 2D en une seule méthode.
- La méthode **matrix()** prend six paramètres, contenant des fonctions mathématiques, ce qui vous permet de faire pivoter, d'écarter, de traduire et d'incliner des éléments.
- Les paramètres sont les suivants:  
**matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())**

```
div {  
  transform: matrix(1, -0.3, 0, 1, 0, 0);  
}
```



- Avec CSS3, On peut ajouter une ombre au texte et aux éléments.
- Ces effets sont réalisés avec les deux propriétés suivantes :

### 1- Les effets d'ombre pour les textes

- La propriété **text-shadow** applique de l'ombre au texte.
- On spécifie l'ombre horizontal et l'ombre vertical shadow.
- On peut spécifier une couleur pour l'ombre.
- On peut aussi ajouter un effet de flou pour l'ombre.
- On peut appliquer plusieurs ombres à un même texte, séparés par des virgules.

- **text-shadow**
- **box-shadow**

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

**Texte avec ombre**

```
h1 {  
    text-shadow: 4px 3px red;  
}
```

**Texte avec ombre**

```
h1 {  
    text-shadow: 4px 3px 6px red;  
}
```

**Texte avec ombre**

```
h1 {  
    text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;  
}
```

**Texte avec 2 ombres**

- La propriété **box-shadow** applique de l'ombre à un élément.
- On spécifie l'ombre horizontal et l'ombre vertical shadow.
- On peut spécifier une couleur pour l'ombre.
- On peut aussi ajouter un effet de flou pour l'ombre.

```
div {  
  box-shadow: 10px 10px 5px grey;  
}
```

Cet élément a un effet d'ombre

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
div {  
  width: 300px;  
  height: 100px;  
  padding: 15px;  
  background-color: coral;  
  box-shadow: 10px 10px 5px 12px  
lightblue;  
}
```

```
</style>  
</head>  
<body>  
<h1>The box-shadow  
Property</h1>  
<div>A div element with a blurred,  
lightblue box-shadow, with a  
spread radius of 12px.</div>  
</body></html>
```

## The box-shadow Property

A div element with a blurred, lightblue box-shadow, with a spread radius of 12px.

- Les animations CSS3 permettent d'animer la majorité des éléments HTML **sans utiliser ni Javascript ni Flash**
- Une animation permet à un élément de changer **progressivement d'un style à l'autre.**
- On peut modifier autant de propriétés CSS que vous le souhaitez, autant de fois que vous le souhaitez.
- Pour utiliser l'animation CSS3, on doit d'abord spécifier des images clés (*keyframes*) pour l'animation.

## Les animations : les images clés

- Les images clés contiennent les styles que l'élément aura à certains moments.
- Lorsque on spécifie des styles CSS dans la règle **@keyframes**, l'animation passe progressivement du style actuel au nouveau style à certains moments.
- Pour que l'animation fonctionne, on **doit lier l'animation à un élément.**

# Les animations : les images clés

- L'exemple suivant lie l'animation "exemple" à l'élément <div>. L'animation durera 4 secondes et changera graduellement la couleur d'arrière-plan de l'élément <div> du rouge au jaune

```
/* code animation */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
/* l'element à appliquer l'animation */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

```

<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  animation-name:
mymove;
  animation-duration: 5s;
}
@keyframes mymove {
  from {left: 0px;}
  to {left: 200px;}
}
</style>
</head>
<body>
<h1>The animation-
name Property</h1>
<p>Specify a name for
the @keyframes
animation.</p>
<div></div>
<p><b>Note:</b> Always
specify the animation-
duration property.
Otherwise the duration is 0,
and the animation will not
be played.</p>
</body>
</html>

```

## The animation-name Property

Specify a name for the @keyframes animation.



**Note:** Always specify the animation-duration property. Otherwise the duration is 0, and the animation will not be played.

## The animation-name Property

Specify a name for the @keyframes animation.



**Note:** Always specify the animation-duration property. Otherwise the duration is 0, and the animation will not be played.



# Les animations : propriétés de l'animation

- Il est aussi possible d'utiliser des pourcentages. Dans ce cas on peut appliquer plusieurs changements de styles.
- L'exemple suivant changera la couleur de l'arrière plan *et la position* de l'élément <div> à 25%, 50% de la durée de l'animation, et lorsque la durée est écoulée : 100%

```
@keyframes example {  
  0%   {background-color:red; left:0px; top:0px;}  
  25%  {background-color:yellow; left:200px; top:0px;}  
  50%  {background-color:blue; left:200px; top:200px;}  
  75%  {background-color:green; left:0px; top:200px;}  
  100% {background-color:red; left:0px; top:0px;}  
}  
  
div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
}
```

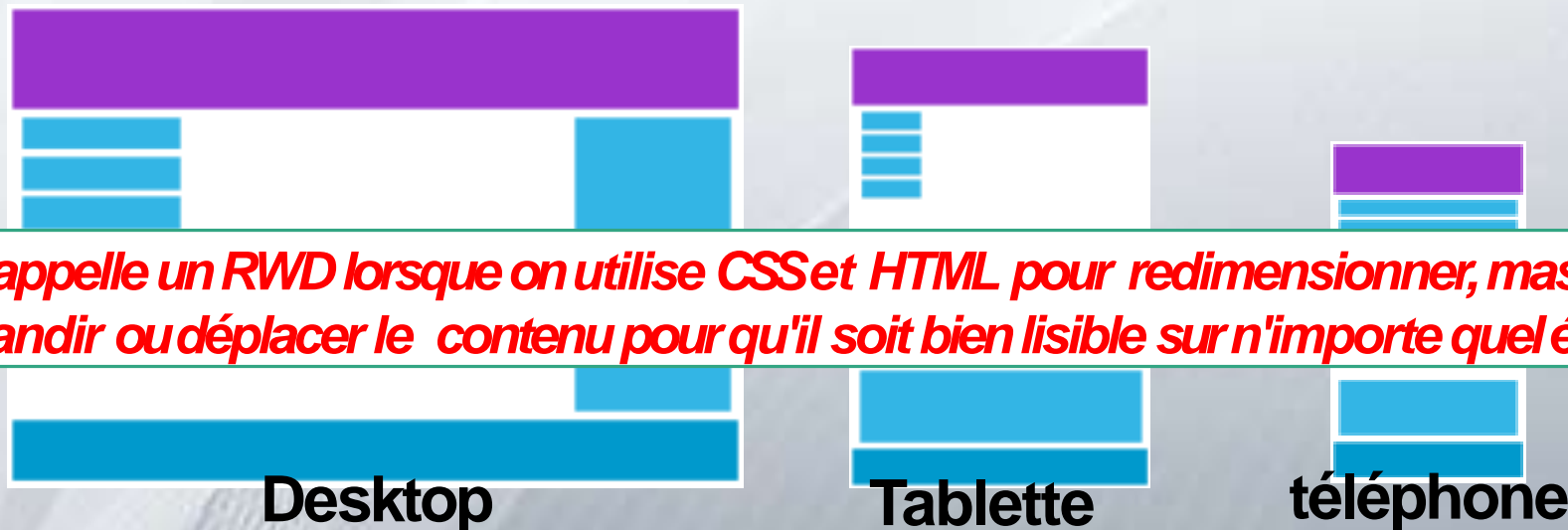
# Les animations : propriétés de l'animation

- Délai de l'animation (**animation-delay**) : cette propriété spécifie le délai pour démarrer l'animation.
- Nombre d'itérations (**animation-iteration-count**) cette propriété spécifie le nombre de fois de l'exécution de l'animation.
  - Si cette propriété prend la valeur "**infinite**" permet de jouer l'animation d'une façon continue
- La courbe de vitesse (**animation-timing-function**) cette propriété spécifie la courbe de vitesse de l'animation.
- Elle peut prendre les valeurs suivantes :
  - **ease** - lent puis rapide , puis lent (par défaut)
  - **linear** la même vitesse du démarrage jusqu'à la fin
  - **ease-in** : démarre lentement
  - **ease-out** finit lentement
  - **ease-in-out** démarre et finit lentement
  - **cubic-bezier(n,n,n,n)** spécifier des valeurs avec la fonction cubic-bezier



- Responsive Web Design (RWD) rend la page Web adaptée à être **bien lisible sur tous les périphériques.**
- RWD utilise uniquement le HTML et le CSS.
- RWD n'est pas un programme ou un JavaScript.
- Les pages Web peuvent être visualisées à l'aide de différents appareils: **ordinateurs de bureau, tablettes et téléphones.**
  - La page Web devrait être bonne et être facile à utiliser, quel que soit le périphérique.

*La même Web peuvent sera visualisée avec différentes dispositions sur différents appareils :*



*Il s'appelle un RWD lorsque on utilise CSS et HTML pour redimensionner, masquer, rétrécir, agrandir ou déplacer le contenu pour qu'il soit bien lisible sur n'importe quel écran.*

# RWD : fenêtre de visualisation (viewport)

- La fenêtre de visualisation (Viewport) est la zone visible de l'utilisateur d'une page Web.
- Le Viewport varie avec l'appareil, et sera plus petite sur un téléphone portable que sur un écran d'ordinateur.
- Avant les tablettes et les téléphones portables, les pages Web étaient conçues uniquement pour les écrans d'ordinateur,
  - Les pages Web aient un design statique et une taille fixe.
- Ensuite, lorsque on a commencé à naviguer à l'aide de tablettes et de téléphones, les pages Web de taille fixe étaient trop grandes pour s'adapter à la fenêtre de visualisation.
  - ➔ ***Pour résoudre ce problème, les navigateurs de ces appareils ont réduit l'intégralité de la page Web pour s'adapter à l'écran.***

# RWD :initialiser la fenêtre de visualisation (viewport)

- Pour contrôler la fenêtre de visualisation, il faut inclure l'élément `<meta>` dans toutes vos pages Web:

```
<meta name="viewport" content="width=device-width,  
initial-scale=1.0">
```

- L'élément `<meta>` donne les instructions au navigateur sur la façon de contrôler les dimensions et la mise à l'échelle de la page.
- **width=device-width** : définit la largeur de la page pour suivre la largeur de l'écran du périphérique (qui variera en fonction de l'appareil).
- **initial-scale=1.0** : définit le niveau de zoom initial lorsque la page est chargée par le navigateur.

- Quelques règles :

1. **NE PAS UTILISER des grands éléments de largeur fixe**

- N'oubliez pas d'ajuster ce contenu pour qu'il corresponde à la largeur de la fenêtre.

2. **NE PAS laisser le contenu s'appuyer sur une largeur de fenêtre particulière pour bien fonctionner.**

- Étant donné que les dimensions de l'écran varient considérablement d'un périphérique, le contenu ne doit pas dépendre d'une largeur de fenêtre particulière.

3. **Utilisez des requêtes de média CSS** pour appliquer un style différent pour les petits et grands écrans.

- Utiliser des valeurs de largeur relatives, (100% par exemple).
- Faire attention à l'utilisation de grandes valeurs de positionnement absolu. Cela peut entraîner l'exclusion de l'élément de la fenêtre sur les petits appareils.

# RWD : requêtes de média CSS

- La requête de média (Media query) est une nouvelle technique de CSS3.
- Il utilise la règle **@media** pour inclure un bloc de propriétés CSS uniquement si une certaine condition est vraie.

## Exemple

- Si la fenêtre du navigateur est inférieure à 500px, la couleur de fond changera en bleu clair(lightblue)

```
@media only screen and
(max-width: 500px) {
  body {
    background-color:
    lightblue;
  }
}
```

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport"
  content="width=device-width, initial-
  scale=1.0">
  <style>
    body {
      background-color: lightgreen;
    }
  </style>
  @media only screen and (max-width:
  500px) {
    body {
      background-color: lightblue;
    }
  }
</html>
```

</head>  
<body>  
<Redimensionnez la fenêtre du navigateur. Lorsque la largeur de ce document est inférieure à 500 pixels, la couleur de fond est "lightblue", sinon il est "lightgreen"..</p>  
</body>

# RWD : CSSMedia queryBreakpoint

- Avec les requêtes de media, on peut ajouter un point d'arrêt **Breakpoint** où certaines parties de la conception se comportent différemment de chaque côté du **Breakpoint**.

## → Astuce : Design mobile first !

- Cela signifie concevoir pour mobile avant de concevoir pour le desktop
  - Cela rendra la page plus rapide sur les petits appareils).
- Au lieu de changer de style lorsque la largeur est inférieure à 768px, nous devrions changer la conception lorsque la largeur est supérieure à 768px. Cela rendra la conception mobile d'abord:

## Exemple

```
/* For mobile phones: */  
[class*="col-"] {  
    width: 100%;  
}  
@media only screen and (min-width: 768px) {  
    /* For desktop: */  
    .col-1 {width: 8.33%;}  
    .col-2 {width: 16.66%;}  
    .col-3 {width: 25%;}  
    .col-4 {width: 33.33%;}  
    .col-5 {width: 41.66%;}  
    .col-6 {width: 50%;}  
    .col-7 {width: 58.33%;}  
}
```

# RWD : CSSMedia query

## Orientation: Portrait / Paysage

- Les requêtes de médias peuvent également être utilisées pour modifier la disposition d'une page en fonction de l'orientation du navigateur.
- On peut avoir un ensemble de propriétés CSS qui ne s'appliqueront que lorsque la fenêtre du navigateur est plus large que sa hauteur, une orientation dite «**Paysage**»:

```
@media only screen and (orientation:
landscape) {
    body {
        background-color: lightblue;
    }
}
```



# RWD : les images

- Utiliser la propriété **width**
  - Si la valeur de **width** est fixée à 100%, L'image sera adaptée et la taille sera modifiable au changement de dimensions du navigateur:
- Utilisation de la propriété **max-width**
  - Dans l'exemple ci-dessus, l'image peut être étendue et être supérieure à sa taille d'origine.
  - ➔ La meilleure solution est d'utiliser la propriété **max-width** à la place de **width**
- Si la propriété **max-width** est fixée à 100%, l'image pourra être réduite, mais ne peut jamais être supérieure à sa taille d'origine:

```
img {  
  width: 100%;  
  height: auto;  
}
```

```
img {  
  max-width: 100%;  
  height: auto;  
}
```

## RWD : les images d'arrière-plan

- Les images d'arrière-plan peuvent également répondre au redimensionnement et à la mise à l'échelle.
- On distingue 3 cas possibles :



1. Si la valeur de la propriété **background-size** est fixée à "**contain**", l'image d'arrière-plan sera redimensionnée et essayer d'adapter la zone de contenu.
  - ➔ Cependant, l'image conservera son ratio (la relation proportionnelle entre la largeur et l'hauteur)

```
div {  
  width: 100%;  
  height: 400px;  
  background-  
image: url('img_flowers.jpg');  
  background-repeat: no-repeat;  
  background-size: contain;  
  border: 1px solid red;  
}
```



2. Si la valeur de la propriété **background-size** est fixée à "**100% 100%**", l'image d'arrière-plan s'étendra pour couvrir toute la zone de contenu:

```
div {  
    width: 100%;  
    height: 400px;  
    background-  
image: url('img_flowers.jpg');  
    background-size: 100%100%;  
    border: 1px solid red;  
}
```



3. Si la valeur de la propriété **background-size** est fixée à « **cover** », L'image d'arrière sera redimensionnée pour couvrir toute la zone de contenu.

- Il est à noter que la valeur « cover » **conserve le ratio d'aspect**, et une partie de l'image d'arrière-plan peut être coupée:

```
div {  
  width: 100%;  
  height: 400px;  
  background-  
image: url('img_flowers.jpg');  
  background-size: cover;  
  border: 1px solid red;  
}
```



- ➔ **Utiliser différentes images pour différents appareils**
- Une grande image peut être parfaite sur un écran d'ordinateur, mais inutile sur un petit appareil (téléphone).
- On peut utiliser des requêtes média pour afficher différentes images sur différents appareils.

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport"
content="width=device-width, initial-
scale=1.0">
<style>
/* For width smaller than 400px: */
body {
  background-repeat: no-repeat;
  background-image: url('img_smallflower.jpg');
}
/* For width 400px and larger: */
```

```
@media only screen and (min-width: 400px){
  body {
    background-image: url('img_flowers.jpg');
  }
}
</style>
</head>
<body>
<p>Rdimensionner le navigateur pour voir
comment l'image d'arrière plan change à partir
de 400px.</p>
</body>
</html>
```