CSS3 Avancé

I. LES DIVISIONS

Principe

Les divisions (ou div) servent à partitionner visuellement une page, en l'organisant en « blocs»: par exemple, des zones de titre, de menu, des colonnes…
Une div est une balise HTML : … englobant une portion de code.
Par exemple, si ma div doit être une zone de menu, elle englobe tous les liens de menu.
La div toute seule n'a aucun effet visible : elle doit absolument être liée à une feuille de style.
Pour cette raison, toutes les div sont nommées. On peut donc réutiliser plusieurs fois la même balise div.

Deux attributs pour les div

Afin de nommer une div, on lui donne un attribut et un nom : … .
Il existe deux attributs : id et class

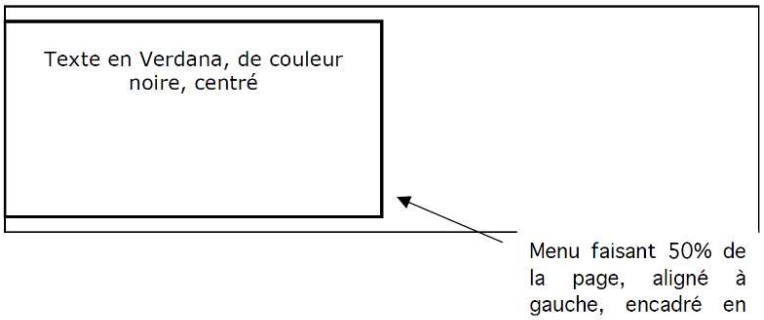
Balise	Code CSS correspondant
`<div id= "nom">`	`#nom{propriete :valeur ;}`
`<div class= "nom">`	`.nom{propriete :valeur ;}`

Exemple :
une doit toujours avoir une largeur (propriété « width ») et un « float » (bord contre lequel s'appuyer)

Pour le code suivant :

```
#menu{
width : 50% ;
float :left ;
background-color :grey ;
color :black;
font-family :Verdana ;
text-align :center ;
border :solid 3px black ;
}
```

On obtiendra à l'écran :



CSS et préfixes de navigateurs

Les nouvelles propriétés CSS3 mettent du temps à être normalisées, c'est donc pour cela que le W3C recommande aux navigateurs web d'utiliser des propriétés "propriétaires" leur permettant d'ajouter les nouvelles fonctionnalités du CSS3 mais aussi de faire des essais en situation réelle avant l'implémentation des propriétés finales.
Elles se distinguent par un préfixe propre à chaque navigateur :

- `-moz-` Mozilla
- `-webkit-` Chrome, Safari, Android...
- `-khtml-` Konqueror
- `-o-` Opera
- `-ms-` Internet Explorer

Afin de donner des angles ronds à une div, vous pouvez utiliser la propriété -moz-border-radius associées à des valeurs chiffrées.

```
.div.arrondi
{
        width:450px;
        background:#fff;
        margin-bottom:3em;
        margin-top:3em;
        padding:0.5em 0;
        -moz-border-radius:10px;
        -webkit-border-radius:10px;
        border-radius:10px;

}
```

Exemple : voir exemple css3_2.html

II. Gérer la transparence

La transparence

La gestion des transparences fait appel à trois propriétés CSS particulières aux navigateurs :

```
-moz-opacity:0.5; (Mozilla (<= 1.6)
opacity: 0.5; (mozilla firefox )
filter:alpha(opacity=50); (IE)
```

Remarque :

Théoriquement, toute la descendance hérite de la propriété de transparence. Il faudra probablement « empiler » deux div pour ne pas rendre transparent tous les contenus de la div avec un fond transparent.

Exemple1 :

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  opacity: 0.5;
  filter: alpha(opacity=50); /* For IE8 and earlier */
}
</style>
</head>
<body>
<h1>transparence d'une image</h1>
<img src="img_forest.jpg" alt="Forest" width="170" height="100"></body></html>
```

Exemple 2:

Sans transparence	Avec transparence

Code de la démonstration

Code HTML:

```
<div class="box" style="width:230px;height:200px">
<img class="imgback" src="images/imgback.jpg" width="230px" height="200px">
<img class="imgtop" src="images/imgtop.jpg" width="230px" height="200px">
</div>
```

Code CSS:

```
.box
{
 position:relative;
}
.imgtop, .imgback
{
 position:absolute;
```

```
  top:0;
  left:0;
}
.imgtop
{
  z-index:10;
  opacity:0.5;
  filter:alpha(opacity=50);
}
```

Pour superposer les images, on leur attribue une position absolue dans un conteneur qui doit lui-même avoir une position relative.

La propriété *z-index:10* place l'image concernée au-dessus de l'autre image.

III. Transparence des couleurs

Le module de couleurs de CSS3 **introduit la notion de transparence dans les valeurs associées à une couleur, l'écriture RGBa**.

Cette composante de la couleur permet de jouer sur les effets d'opacité entre les différentes couches d'éléments HTML.

La notation RGBa obéit aux mêmes règles de fonctionnement que la notation classique RGB, mis à part qu'une composante est ajoutée à la valeur : rgb(0,0,0) devient donc **rgba(0,0,0,0)**. La dernière valeur (l'alpha) indiquant le degré d'opacité entre 0 et 1.

Exemple :

Illustrons cette notation sur une structure HTML, l'objectif étant d'appliquer une transparence de 50% sur la couleur d'arrière-plan de l'élément

```
<div>
  <h1>Joli titre</h1>
</div>
```

Voici les styles CSS permettant d'appliquer cette transparence :

```
div {
  background-color: rgba(0, 0, 255, 0.5); /* un arrière plan bleu à 50% de transparence */
}
div h1 {
  color: rgb(255, 200, 0); /* une couleur de texte jaune-orange */}
```

Pour s'assurer qu'il y ait une **alternative** pour les mauvais élèves ou les navigateurs plus anciens, il est possible de commencer par déclarer une couleur de fond solide à l'aide de l'écriture classique RGB. Les anciens navigateurs, pour la plupart, ne reconnaîtront pas la seconde valeur et se contenteront de la première déclaration :

```
div {
  background-color: rgb(0, 0, 255);
  background-color: rgba(0, 0, 255, 0.5);
}
```

Exemple

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <style type="text/css">
div {
  background-color: rgba(0, 0, 255, 0.5);
}
    </style>
    <title>Drgba</title>
  </head>
  <body>
    <p>Exemples d'utilisations pour CSS3</p>
  <div>
    <p>mon premier paragraphe</p>
    <p>mon deuxième paragraphe</p>
  </div>  </body></html>
```

IV. Les transformations

Les transformations permettent d'agir sur les éléments de la page en les déformants selon des formules géométriques bien connues : translation, rotation, inclinaison... Ces transformations ont souvent peu d'intérêt dans les mises en page qui restent en général très rectangulaires, mais prennent tout leur sens dans des animations en offrant des possibilités graphiques nouvelles.

Les transformations vont toutes être effectuées par l'utilisation de la propriété **transform** qui prendra en valeur l'effet de transformation souhaité et son taux de transformation de la façon suivante :

```
transform : type_de_transformation(valeur);
```

On pourra cumuler les transformations en les listant, séparées par un espace :

```
transform : type_de_transformation(valeur)  type_de_transformation(valeur);
```

Chaque transformation possède un mot-clé générique et une variante qui précise l'axe de transformation avec X, Y ou Z.

Rotation en CSS

La rotation sera basé sur la transformation identifié par le mot clé **rotate**. Mais il existe d'autres transformations **rotateX**, **rotateY** et éventuellement **rotateZ**.

```
p#id2{
  -webkit-transform: rotateY(0deg);
  -webkit-animation-name : trans1;
}

@-webkit-keyframes trans1{
  from {
    -webkit-transform:rotateY(0deg)
  }
  to {
    -webkit-transform:rotateY(360deg)
  }
}
```

La rotation, comme on peut la voir est exprimée en degré, **deg**.

La règle CSS @keyframes

La règle CSS ***@keyframes*** permet de regrouper les images clés concernant une animation CSS3.

Exemple de syntaxe CSS @keyframes{} :

```
@keyframes identifiant{
```

```
from {width: 300px}
to {width: 100px;}        }
```

Mise à l'échelle en CSS

La mise à l'échelle ou homothétie permet de changer la dimension des éléments en rapport avec leur taille actuelle, c'est-à-dire sans changer les valeurs appliquées à width et height.

```
p#id1{
 -webkit-transform: scale(1);
 -webkit-animation-name : trans2;
}

@-webkit-keyframes trans2{
 from{-webkit-transform:scale(1)}
 to{-webkit-transform:scale(0.5)}
}
```

L'échelle est par défaut exprimé entre valeur de 1 (taille actuelle). Ce qui est inférieur à 1 réduit la taille et ce qui supérieur l'augmente selon un produit :

taille actuelle * scale = taille finale.

Translation

La translation permet de faire des déplacements dans la page en spécifiant la distance à parcourir au lieu de définir la nouvelle position comme il faudrait le faire avec les propriétés top, left ou encore margin-top margin-left.

Exemple :

```
<!doctype html>

<html>
<head>
<title>Exemples CSS3</title>

<style>
p {
 border: solid red;

 -webkit-transform: translate(100px) rotate(20deg);
 -webkit-transform-origin: 0 -250px;

 transform: translate(100px) rotate(20deg);
 transform-origin: 0 -250px;
}
</style>
```

```
</head>
<body>

<p>L'élément transformé</p>

</body>
</html>
```

L'inclinaison

La dernière des transformations est l'inclinaison.

```
p#id4{
  -webkit-transform: skew(0);
  -webkit-animation-name : trans4;
}

@-webkit-keyframes trans4{
  from{}
  to{-webkit-transform:skew(45deg)}
}
```

On a la possibilité de mettre deux valeurs pour définir les deux sens de déformation sous la forme **skew(20deg 45deg)** ou encore en passant séparément par **skewX** et **skewY**.

Ordre d'application des transformations

Il est important de bien choisir l'ordre dans lequel les transformations seront appelées car le résultat peut changer radicalement. Par exemple:

```
p {
  background : red;
  width:100px;
  height:100px;
}
#id1 {
  -webkit-transform: skew(45deg) rotate(45deg)
}

#id2 {
  -webkit-transform: rotate(45deg) skew(45deg)
}
```

Exemple transformation :

```
<!DOCTYPE html>
<html>
```

```
 <head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <style type="text/css">
div{
 width: 50px;
 height: 50px;
 background-color: orange;
 border: 1px solid black;
 box-sizing: border-box;
 margin: 20px;
}
.d1{
 transform-origin: 0 0;
 transform: translate(100px);
}
.d2{
 transform-origin: 50% 50%;
 transform: translate(100px);
}
.d3{
 transform-origin: 0 0;
 transform: rotate(45deg);
}
.d4{
 transform-origin: 50% 50%;
 transform: rotate(45deg);
}
   </style>
   <title>Drgba</title>
 </head>
 <body>
 <div></div>
<div class="d1"></div>
<div class="d2"></div>
<div class="d3"></div>
<div class="d4"></div>
 </body>
</html>
```

Exemple : Effets 3D avec les transformations 2D

Un bon usage des transformations 2D peut aboutir à des effets de dessin en 3D comme par exemple ce cube.

```
<body>
 <p id="id1">ID1</p>
 <p id="id2">ID2</p>
 <p id="id3">ID3</p>
```

```
</body>


 p {
  position: absolute;
  margin-left:100px;
  height: 100px;
  width: 100px;
}

#id1 {
  background: #9acc53;
  -webkit-transform:
    rotate(-45deg)
    skew(15deg, 15deg);
}

#id2 {
  background: #8ec63f;
  -webkit-transform:
    rotate(15deg)
    skew(15deg, 15deg)
    translate(-50%, 100%);
}

#id3 {
  background: #80b239;
  -webkit-transform:
    rotate(-15deg)
    skew(-15deg, -15deg)
    translate(50%, 100%);
}

#id3:hover {
  -webkit-animation-name:porte;
  -webkit-animation-duration:2s;
}

@-webkit-keyframes porte{
  from {}
  to { -webkit-transform: rotate(-15deg) skew(-15deg, -15deg) translate(-50%, 100%)}
}
```

V. Transition

Les transitions CSS permettent de modifier partiellement un objet pour qu'il passe d'un état à un autre en jouant sur la valeur de ses propriétés CSS actuelles. Les transitions CSS seront particulièrement utilisées pour placer une rétroaction visuelle au survol d'un bouton par exemple.

Les transitions sont donc plus simples à mettre en place que les animations qui permettront de jouer sur des durées, modifications et étapes plus variées, mais elles sont aussi beaucoup plus simples à mettre en oeuvre en ce qu'elles sont réellement, au niveau du code, très proche des autres propriétés CSS. Elles s'effectueront par le biais de 4 propriétés principales

- *transition-property*
- *transition-duration*
- *transition-timing-function*
- *transition-delay*

Une transition est une animation basique qui fait passer un objet d'un état A à un état B. Par exemple, on peut dire : "Déplace cet objet de la gauche vers la droite de l'écran tout en changeant sa couleur". L'image que vous venez de voir représente une transition : tout bouge en même temps.

Exemple

```
<!DOCTYPE html>
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
div{
  width: 90%;
  height: 100px;
  margin: 20px auto;
  padding: 40px;
  border: 2px solid black;
  box-sizing: border-box;
  background-color: orange;
}
.d1{
  transition-property: background-color;
  transition-duration: 5s;
```

```
}
.d2{
    transition-property: background-color, border;
    transition-duration: 5s;
}
.d3{
    transition-property: all;
    transition-duration: 5s;
}
.d1:hover, .d2:hover, .d3:hover{
    background-color: blue;
    border: 5px solid red;
    color: white;
}
    </style>
    <title>Drgba</title>
  </head>
  <body>
 <div class="d1">Du texte dans le div 1</div>
<div class="d2">Du texte dans le div 2</div>
<div class="d3">Du texte dans le div 3</div>
  </body>
</html>
```

VI. Les animations CSS3.

Les étapes de création d'une animation :

- **Déclarer une animation CSS3.**

En Css3 une animation est regroupée dans une règle Css **@keyframes**.

Cette règle décrit les propriétés CSS à modifier **par image clé**.

L'image clé correspond à un moment de votre animation Css et elle est identifiée par un pourcentage où **0%** correspond à la première image clé (obligatoire) et **100%** à votre dernière image clé (obligatoire).

Si la durée est de 20s (voir animation-duration) alors 0% correspond à la seconde 0 et 100% à la seconde 20 de l'animation Css3.

il faut faire quatre déclarations :

```
@-webkit-keyframes nomanimation { ... } /* -webkit- Chrome et Safari */
@-moz-keyframes nomanimation { ... } /* -moz -Firefox */
@-o-keyframes nomanimation{ ... } /* -o- Opera */
@keyframes nomanimation{ ... } /* Navigateur aux normes w3c */
```

- **Paramétrer votre animation CSS3.**

Maintenant que nous avons décrit en CSS3 le comportement de notre animation grâce à la règle CSS3 **@keyframes**, il nous reste quelques paramétrages à faire.

animation-name, propriété d'animation de nommage

La propriété Css3 ***animation-name*** permet de spécifier le nom de l'animation à utiliser.

```
@keyframes nomanimation{ ... }
div.bloccible{ animation-name : nomanimation; }
```

animation-duration, propriété d'animation de durée

Cette durée peut être exprimée en :

- **secondes**, le nombre sera suivit par "**s**"

- **millisecondes** : le nombre sera suivit par "**ms**".

```
@keyframes nomanimation { ... }

div.bloccible{ animation-name : nomanimation; animation-duration : 20s; }
```

animation-iteration-count, propriété d'animation de répétition

Permet de préciser le nombre de fois que l'animation Css3 sera jouée. La propriété css3 ***animation-iteration-count:infinite*** va permettre de jouer en boucle l'animation CSS3.

La valeur par défaut est 1.

```
@keyframes nomanimation { ... }
```

```
div.bloccible{ animation-name : nomanimation; animation-iteration-count : infinite; }
```

animation-fill-mode

animation-fill-mode permet de spécifier la manière dont doit être interprété les propriétés css contenues dans la première ou dernière image clé.

Exemple :
Dans le code suivant, la couleur initiale est yellow, quand l'animation Css va commencer la couleur devient red puis évolue vers le green et une fois l'animation Css finie la couleur redevient yellow,

```
@keyframes nomanimation
 { 0%{background-color:red;}
100%{background-color:green;}                                                    }
div.bloccible{ animation-name : nomanimation; background-color:yellow; }
```

Dans le code suivant, la couleur initiale est la couleur de la première image clé, quand l'animation Css3 va commencer la couleur évolue vers le green et une fois l'animation Css3 finie la couleur reste green,

```
@keyframes nomanimation { 0%{background-color:red;} 100%{background-color:green;} }
div.bloccible{ animation-name : nomanimation; background-color:yellow; animation-fill-mode:both;}
```

Les autres propriétés d'animation CSS3.

Il existe d'autres propriétés CSS3 pour paramétrer votre animation Css que je vous laisse découvrir :

- ***animation-timing-function*** : cette propriété css3 permet de spécifier la manière de progresser entre les images clés. Sa valeur par défaut est **ease**.

- ***animation-play-state*** : cette propriété css3 permet de mettre en pause l'animation Css. Sa valeur par défaut est **running**.

- ***animation-direction*** : cette propriété css3 permet de spécifier le sens de lecture de l'animation. Sa valeur par défaut est **normal**

- ***animation-delay*** : cette propriété css3 permet de spécifier un temps de latence avant de commencer la lecture de l'animation. Sa valeur par défaut est **0s**.

- ***animation-direction*** : cette propriété css3 permet de spécifier permet de spécifier le sens de lecture de l'animation appliquée, notamment pour les répétitions. Sa valeur par défaut est **normal**.

Déclencher une animation CSS3.

- Directement à l'exécution
- Au changement d'état.
- Assignation d'une classe
-

Vous pouvez déclencher votre animation CSS3 lors d'un changement d'état par exemple lorsque la souris survole l'élément qui doit être animé. Pour cela vous devez utilise une ***pseudo-classe*** comme ***:hover***, ***:focus***,.....

Exemple : Assignation d'une classe

Une autre possibilité pour déclencher votre animation CSS3, consiste à assigner en Javascript la classe à l'élément qui doit être animé.

```
<style>@keyframes monanimation {
0% {margin-left:0;}
100% {margin-left:100px;}
}
.mon-animation{ animation-name : nomanimation;}</style>
<div id="bloc-anime" onclick="addAnimation()">element animable</div>
<script>
function addAnimation(){
// Avec Jquery
$("#bloc-anime").addClass("mon-animation")
// En javascript
document.getElementById("bloc-anime").classList.add("mon-animation")
}
</script>
```

Animation vs transition.

En règle générale, vous utiliserez plus souvent des transitions (***transition***), car votre "Animation" se limite à un changement ne nécessitant que deux étapes.

Partez du principe que vous devez utiliser les transitions par défaut. Voici les éléments qui font que vous utilisez la propriété css ***@keyframes pl***utôt que la propriété css ***transition*** :
- L'Animation doit se répéter.

- L'Animation a besoin de points intermédiaires.

- L'Animation démarre toute seule.

- L'Animation peut faire une pause.

Sachez aussi :

- que les *Transitions* sont plus simples à gérer en javascript.

- que lorsqu'une transition est interrompue durant son exécution, chaque propriété modifiée peut revenir à son état initial en étant animées, ce que ne peut pas faire les animations.