Apport du CSS3

Avec l'avènement du CSS3, il devient plus simple de réaliser des pages web ergonomiques, élégantes et de créer des effets et des animations sans avoir recours à un langage de programmation.

1. Couleurs et images

a. Couleurs

La propriété `opacity` permet de définir le degré d'opacité ou de transparence d'un élément donné.

Les valeurs attendues sont comprises entre 0 et 1, sachant que 1 signifie qu'il n'y a pas de transparence, que 0 signifie une transparence totale et que 0.5 représente une transparence à 50 %...

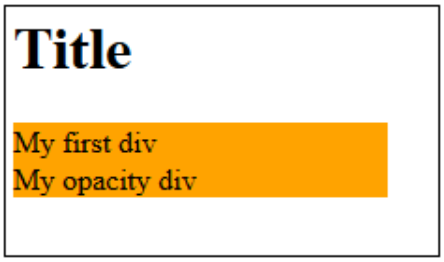
Voici un exemple d'utilisation :

```
<!DOCTYPE html>
<html>

  <head>
    <style>
   div{
      background-color: orange;
        width: 200px;
   }

   div#opacity{
       opacity: 0.2;
    }

    </style>
  </head>
  <body>
    <h1>Title</h1>
    <div>My first div</div>
    <div id="opacity">My opacity div</div>
  </body>
</html>
```



La propriété `rgba` permet de spécifier une couleur, plus un degré d'opacité.

Comme pour la fonction `rgb()`, `rgba()` utilise les valeurs rouge, vert et bleu d'une couleur et ajoute l'opacité.

Voici un exemple :

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

La dernière valeur spécifie le degré d'opacité.

Le principe de la transparence RGBa est radicalement différent de la propriété `opacity`. En effet, `opacity` s'applique à l'élément et aussi à tous ses descendants alors que RGBa s'applique uniquement à l'élément sélectionné.

b. Images

Avec CSS3, il est possible désormais de déclarer plusieurs images de fond avec la propriété `background-image`. C'est la première déclarée qui se trouvera en premier plan, les autres se placeront dessous.

Voici la syntaxe :

```
background-image: url(image1.png), url(image2.png), url(image3.png);
```

Dans le cas d'images multiples, si on veut utiliser la propriété `background-repeat` (pour spécifier si l'image de fond doit se répéter ou non) ou la propriété `background-position` (qui spécifie le placement de l'image de fond). Il faudra répéter les valeurs autant de fois qu'il y a d'images de fond.

Voici un exemple concret :

```
p{
    background-image: url(image1.png), url(image2.png), url(image3.png);
    background-repeat: no repeat, repeat, no repeat;
    background-position: center left, left top, center right;
}
```

CSS3 apporte trois nouvelles propriétés liées à l'image de fond :

- `background-origin` qui correspond à la position d'origine d'une image.
- `background-clip` qui permet de rogner une image de fond.
- `background-size` qui permet de définir la taille de l'image de fond par rapport à son contenant.

2. Effet de texte, bordures et dimensions

a. Effet de texte, police web

La propriété `text-shadow` permet d'appliquer des effets à du texte, dont particulièrement l'ajout d'une ombre portée.

La syntaxe à utiliser est la suivante :

```
text-shadow: h-pos v-pos (blur) (color);
```

Les valeurs à spécifier sont :

- `h-pos`, pour la position horizontale de l'ombre.
- `V-pos`, pour la position verticale de l'ombre.
- `Blur`, pour la distance du flou.
- `Color`, pour la couleur de l'ombre.

Voici un exemple :

```
h3{
      text-shadow: 1px 2px 3px #333;
}
```

Il est possible d'intégrer dans une page web des polices exotiques comme les Google Fonts qui sont hébergées sur les serveurs de Google. Pour utiliser l'une des polices mises à disposition par Google ou un autre fournisseur de polices en ligne, voici comment procéder :

- Il faut ajouter la balise `link` avec le lien vers la font-style hébergée chez le fournisseur de polices dans la page web HTML.
- Appliquer cette police à une classe CSS avec la propriété `font-family`.

Voici un exemple :

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href=
"http://fonts.googleapis.com/css?family=Tangerine">
    <style>
      p {
        font-family: 'Tangerine', serif;
        font-size: 28px;
      }
    </style>
  </head>
  <body>
    <h1>Title</h1>
    <p>My first paragraph</p>
    <span>this is a span</span>
  </body>
</html
```

b. Bordures

Il est désormais possible de réaliser facilement des bordures à coins arrondis grâce à la propriété `border-radius`.

Celle-ci permet d'arrondir les angles de n'importe quel élément en spécifiant la taille de l'arrondi en pixels.

Voici un exemple :

```
p{
    border-radius: 5px;
}
```

c. Dimensions

Lorsqu'on utilise `width` et `height` pour fixer respectivement la largeur et la hauteur d'un bloc, cela ne prend pas en compte la taille des marges et l'épaisseur des bordures. Donc, pour connaître une largeur finale d'un bloc, il faut additionner la largeur aux marges et aux épaisseurs des bordures.

Ce que propose la propriété `box-sizing`, c'est de fixer hauteur et largeur tout compris avec la valeur `border-box`. Seules les marges externes ne sont pas prises en compte. Après, il suffit de spécifier comme habituellement la largeur du bloc avec `width` et la hauteur avec `height`.

Voici un exemple :

```
div{
      box-sizing: border-box;
      width: 400px;
      height: 150px;
      background-color: orange;
}
```

La propriété `resize` autorise le redimensionnement d'un bloc avec la souris. Pour qu'elle soit effective, il faut également que le bloc dispose de la propriété `overflow` avec la valeur `auto`.

Les valeurs possibles pour la propriété `resize` sont :

- `both` pour spécifier que la hauteur et la largeur sont redimensionnables.

- `vertical` pour spécifier que la hauteur et la largeur sont redimensionnables.
- `horizontal` pour spécifier que la hauteur et la largeur sont redimensionnables.

```
div{
      resize: both;
      overflow: auto;
}
```

d. Débordement de contenu

Les propriétés `overflow-x` et `overflow-y` définissent la façon dont un élément doit traiter un contenu dont les dimensions dépassent la hauteur (y) ou la largeur (x) du contenant.

Les syntaxes à utiliser sont :

```
overflow-x: value;
overflow-y: value;
```

Les valeurs possibles pour les propriétés `overflow-x` et `overflow-y` sont les suivantes :

- `auto` ajoute une barre de défilement si nécessaire.
- `hidden` signifie que tout ce qui dépasse du contenant est caché et qu'il n'y a pas de barre de défilement.
- `scroll` ajoute systématiquement une barre de défilement.
- `visible` signifie que le contenu n'est pas rogné : il est visible et peut dépasser de son contenant.

3. Multicolonnage

Les nouvelles propriétés de multicolonnage évitent désormais l'écriture de nombreuses lignes de code JavaScript.

Pour spécifier un nombre de colonnes fixe, il existe désormais la propriété `column-count`. La valeur attendue est un entier qui désigne le nombre de colonnes.

Voici un exemple :

```
p{
      column-count: 4;
}
```

Pour spécifier la largeur des colonnes, il existe désormais la propriété `column-width`. La valeur attendue est une taille en px, em ou %.

Voici un exemple :

```
p{
      column-width: 12em;
}
```

Il existe d'autres propriétés qui permettent de spécifier certains paramètres de multicolonnage :

- `column-gap` pour définir l'espacement entre les colonnes.
- `column-rule-style`, `column-rule-width`, `column-rule-color` pour spécifier respectivement le style du trait de séparation entre les colonnes, son épaisseur et sa couleur.

4. Animations, transitions et transformations

a. Animations

La règle `keyframes` permet de réaliser des animations avec les éléments HTML tels que des images ou du texte. Il est possible avec `keyframes` de réaliser un diaporama sans utiliser de JavaScript tout en CSS3.

La syntaxe standard est la suivante : le mot-clé `@keyframes`, puis le nom de l'animation et entre les accolades un point de départ à définir avec `from` et le point d'arrivée avec `to` :

```
@keyframes animation_name {
      from {
            //style CSS
      }
      to {
            //style CSS
      }
}
```

Ensuite, il faut définir une règle CSS pour l'élément que l'on souhaite animer et appeler avec la propriété `animation` la règle `keyframes` que l'on a déclarée précédemment :

```
element{
      animation  animation_name time;
}
```

`time` désigne en secondes le temps que va durer l'animation.

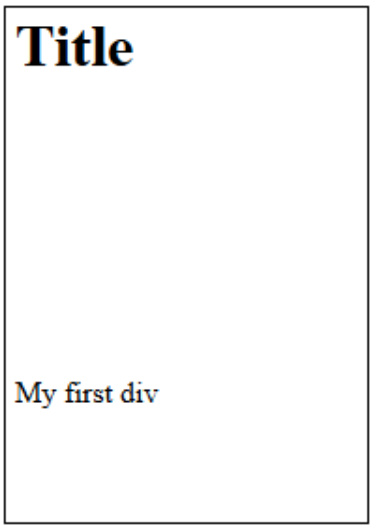
Voici un exemple d'utilisation de la règle keyframes : on a un élément de type `div` qui contient du texte et on souhaite faire bouger ce texte vers le bas. On utilisera les propriétés `margin-top` pour faire bouger le texte du haut vers le bas :

```
<html>
  <head>
    <link rel="stylesheet" type="text/css"
href="http://fonts.googleapis.com/css?family=Tangerine">
    <style>
    @keyframes textmovement {
      from {
            margin-top:20px;
      }
      to {
            margin-top:220px;
```

```
      }
     }

div{
      animation: textmovement 5s;
}


    </style>
  </head>
  <body>
    <h1>Title</h1>
    <div>My first div</div>
  </body>
</html>
```



b. Transitions

Les transitions permettent de modifier l'aspect visuel d'un texte ou d'un bloc : changement de taille, de position, de forme, de couleurs…

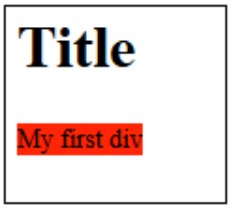
Pour mettre en place des transitions, il faut utiliser la propriété `transition`.

Voici un exemple de transition sur un élément de type `a` qui permet de changer progressivement durant 5 secondes la couleur de fond du lien lors du survol avec la souris :

```
<html>
  <head>
```

```
    <style>
     a{
           background-color: yellow;
           font-size:14px;
           transition: background-color 5s;
           width: 200px;
     }
     a:hover{
           background-color:red;

     }
    </style>
  </head>
  <body>
    <h1>Title</h1>
    <a>My first div</a>
  </body>
</html
```



Il existe plusieurs autres propriétés permettant de configurer les animations :

- `transition-property` : si plusieurs propriétés sont indiquées, cela permet de leur associer des paramètres de vitesse, durée ou délai distincts.
- `transition-duration` : permet de spécifier la durée de la transition.
- `transition-timing-function` : permet de définir si une animation doit être régulière ou saccadée.
- `transition-delay` : définit la durée d'attente avant de démarrer l'animation.

c. Transformations

CSS3 donne la possibilité de réaliser des transformations 2D et 3D.

Pour utiliser les fonctions de transformation 2D et 3D, il faut au préalable passer par la propriété `transform`.

La syntaxe à utiliser est alors la suivante :

```
transform: function();
```

Parmi les fonctions de transformation 2D, on a :

- `translate(X,Y)` : permet d'effectuer un déplacement, décalage horizontal de x et vertical de y.
- `rotate(angle)` : effectue une rotation de l'angle spécifié en paramètres.

Voici un exemple d'utilisation de la propriété `transform` avec la fonction `translate`, qui effectue un déplacement de la div :

```
<html>
  <head>
    <style>
   div {
       transform: translate(20px, 150px);
       background-color: orange;
       }


    </style>
  </head>
  <body>
    <h1>Title</h1>
    <div>My first div</div>
  </body>
</html>
```

Parmi les fonctions de transformation 3D, on a :

- `translate3d(x,y,z)` : décalage horizontal de x, décalage vertical de y et en profondeur de z.
- `rotate3d(x,y,z,angle)` : rotation de l'angle spécifié autour du vecteur dont les composantes sont x, y et z.

5. Media queries

Avec CSS3, il est désormais possible d'affecter des styles en fonction des terminaux qui vont afficher les contenus.

Voici un exemple simple :

```
@media screen and (max-width : 1200px){
  //règles CSS
}
```

Ici, les propriétés CSS seront appliquées pour un terminal de type écran dont la largeur de fenêtre ne dépassera pas 1200px.

Les media queries seront abordées en détail dans le chapitre Pensez votre application responsive de cet ouvrage.