De la géométrie avec CSS

par debray jerome (Dji programmation web2 et design) (Blog)

Date de publication :

Dernière mise à jour :

Grâce aux CSS et à la nouvelle norme CSS3, nous pouvons créer de plus en plus de formes tels que les carrés, les rectangles, les ronds, etc. Dans cette article, je vais présenter les différentes possibilités de forme faisable en CSS (du moins une liste exhaustive).

Tous les exemples auront pour structure HTML cette base :

```
<div class="nom_de_la_forme">
</div>
```

I - Le carré

Ici, rien de bien compliqué, il suffit de définir une largeur et une hauteur identique pour chaque coté de notre élément HTML.

```
.carre{
 width:200px;
 height:200px;
 background:#069;
}
```

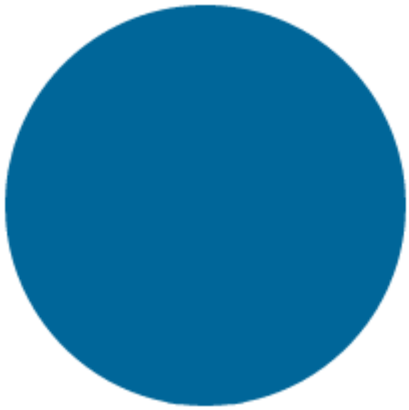
II - Le rectangle

Le principe est le même que le carré sauf que la largeur (ou la hauteur selon l'effet désiré) est plus grande que la hauteur.

```
.rectangle{
 width:400px;
 height:200px;
 background:#069;
}
```

III - Le rond

Le rond est obtenu grâce à la propriété **border-radius** sur un élément carré. La valeur du **border-radius** sera égale à la moitié de la valeur d'un coté.



```
.rond{
 width:200px;
 height:200px;
 background:#069;
 -webkit-border-radius:100px;
 -moz-border-radius:100px;
 -o-border-radius:100px;
 border-radius:100px;
}
```

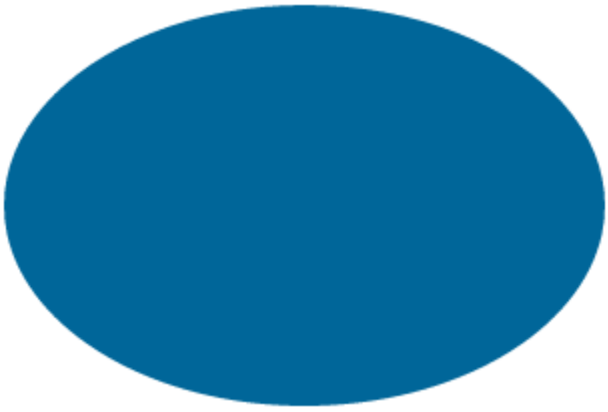
IV - L'ovale

J'ai appris récemment que l'on pouvait donner 2 valeurs au **border-radius** équivalent à l'arrondi de départ et l'arrondi d'arrivée d'un coin de bordure.

Par exemple :

```
border-top-left-radius : 150px 100px;
```

Le code ci dessus va donner pour la bordure haut et droite, un arrondi commançant à 100px en radius sur le left pour arriver à 150px en radius sur le top.



Ainsi en appliquant ce procédé à toutes les bordures, on arrive à une forme ovale. Pour se faire, il faudra le faire sur un bloc ayant une forme rectangle.

```
.ovale{
 width:300px;
 height:200px;
 background:#069;
 -webkit-border-radius:150px / 100px;
 -moz-border-radius:150px / 100px;
 -o-border-radius:150px / 100px;
 border-radius:150px / 100px;
}
```

équivaut à :

```
.ovale {
 width: 300px;
 height: 200px;
 background: #069;
 border-top-left-radius: 150px 100px;
 border-top-right-radius: 150px 100px;
 border-bottom-right-radius: 150px 100px;
 border-bottom-left-radius: 150px 100px;
}
```

V - Les triangles

Le principe du triangle est simple car il équivaut à manipuler la largeur des bordures ainsi que leur visibilité - par le biais de la couleur de la bordure - sur un élément HTML ayant des dimensions égales à 1px.

```
.triangle{
 width:1px;
 height:1px;
 border:1px solid #069;
```

```
 border-color:transparent transparent #069 transparent;
 border-width:100px;
}
```

Pour avoir un triangle orienté vers un autre coté, il suffit juste de changer les valeurs de **border-color**.

VI - La transformation au service de la géométrie : le parrallélogramme

Dans mon article précédent, je parlais des propriétés **transform** introduites en CSS3. Celles ci nous permettent de modifier la forme de nos éléments (dans un plan 2D).

Voici un rappel des principales fonctions de transformation :

- skew
- rotate
- translate

La fonction qui va nous intéresser sera la fonction **skew**. On part de la base d'un rectangle auquel on applique la transformation :

```
.parrallelogramme{
 width:300px;
 height:100px;
 background:#069;
 -webkit-transform:skew(30deg);
 -moz-transform:skew(30deg);
 -o-transform:skew(30deg);
 tranform:skew(30deg);
}
```

VII - Le trapeze

Un trapèze se fait sur la même base que le triangle mais au lieu d'avoir une hauteur ou une largeur à 1px, on a un des 2 cotés à 1px et l'autre ayant une valeur supérieure à 1px (100px par exemple).

```
.trapeze{
 width:1px;
 height:100px;
 border:1px solid #069;
 border-color:transparent transparent transparent #069;
 border-width:100px;
}
```

VIII - Les autres formes : la magie des pseudo-éléments :after et :before

Jusque là, on a vu la puissance du CSS/CSS3 pour produire de la geométrie. Grâce aux pseudo-éléments **:after** et **:before** (ou **::after** et **::before**), on peut concevoir des formes beaucoup plus complexe.

Dans le tutoriel d'Alen Grakalic que j'ai traduit (**tags pour vos billets blogs en css3 pure**), on voit un peu ce que l'on peut obtenir grâce à ces pseudo-éléments. Voyons nous ce que l'on peut en faire.

IX - l'étoile

L'astuce est assez simple, car il suffit juste de créer un triangle. Puis grâce à la pseudo classe **:after**, on refait un triangle mais dans le sens opposé.

Une fois ces 2 triangles créés, il suffit de positionner le triangle créé en **:after** afin de le superposer au deuxième triangle.

```
.etoile{
 margin:100px auto;
 width: 0;
 height: 0;
 border-left: 50px solid transparent;
 border-right: 50px solid transparent;
 border-bottom: 100px solid #069;
 position: relative;
}
.etoile:after{
 content:'';
 border: 1px solid #069;
 border-left: 50px solid transparent;
 border-right: 50px solid transparent;
 border-top: 100px solid #069;
 width:0;
 height:0;
 position: absolute;
 top:30px;
 left:-50px;
}
```

X - Triangle et rond

C'est le même principe que pour l'étoile.

```
.rondTriangle{
 margin:100px auto;
 width: 0;
 height: 0;
 border-left: 50px solid transparent;
 border-right: 50px solid transparent;
 border-bottom: 100px solid #000;
 position: relative;
}
.rondTriangle:after{
 content:'';
 background:#069;
 -webkit-border-radius:50px;
 -moz-border-radius:50px;
 -o-border-radius:50px;
 border-radius:50px;
 width:75px;
 height:75px;
```

```
 position: absolute;
 top:30px;
 left:-37px;
}
```

XI - Le losange

Il s'agit de 2 triangles juxtaposés.

```
.losange{
 margin:100px auto;
 width: 0;
 height: 0;
 border-left: 50px solid transparent;
 border-right: 50px solid transparent;
 border-bottom: 100px solid #069;
 position: relative;
}
.losange:before{
 content:'';
 border: 1px solid #069;
 border-left: 50px solid transparent;
 border-right: 50px solid transparent;
 border-top: 100px solid #069;
 width:0;
 height:0;
 position: absolute;
 top:100px;
 left:-50px;
}
```

On aurait pu faire ce losange en passant par un carré avec 2 fonctions de transformations : **skew** et **rotate**.

XII - Conclusion

A partir de là, on peut imaginer plein de possibilités de formes grâce aux nouvelles propriétés CSS3 et aux pseudo-éléments.

XIII - Remerciements