Les flottements

Comme nous l'avons dit dans la leçon précédente, il existe diverses manières d'effectuer une mise en page. Nous allons nous intéresser à la propriété que l'on a coutume d'appeler le « **flottement** » et dont les équivalents peuvent être le modèle **flexbox**, la propriété **display** et de manière beaucoup plus marginale la propriété **position**. Pourquoi commencer par les flottements alors que ce n'est pas la méthode la plus facile à maîtriser ? Parce que, malgré son âge avancé, c'est également celle qui fonctionne dans presque toutes les situations.

I. Définition des flottements

Élément flottant
Un élément flottant est un élément qui sort du flux naturel et se place à gauche ou à droite.

Le problème que génère un élément flottant est que le parent n'a pas conscience de la taille que vont prendre ses enfants flottants. Pour les éléments qui suivent, on observe également qu'ils vont compenser le flottant pour calculer l'espace qu'ils prennent. C'est pourquoi, après avoir réalisé un flottement, il convient absolument de mettre fin à la flottaison pour remettre les éléments dans le flux normal.

C'est la règle CSS **float** qui permet à un objet HTML de sortir du flux normal d'affichage, tandis que **clear** va supprimer les effets du **float** pour que l'élément retrouve le flux naturel. Nous verrons plus loin d'autres méthodes pour mettre fin à la flottaison.

Concrètement, comment ça fonctionne ?

CSS
float: left;

Un objet HTML sur lequel on applique cette règle se placera à gauche de l'objet qui le suit dans la page HTML.

float: right;

Un objet HTML sur lequel on applique cette règle se placera à droite de l'objet qui le suit dans la page HTML.

clear: both;

Un objet HTML sur lequel on applique cette règle retrouvera le flux normal d'affichage, quel que soit le **float** utilisé auparavant, **right** ou **left**. De plus, le parent de l'élément flottant prendra en compte tous les enfants pour calculer sa propre hauteur.

Il existe également **clear: left** et **clear: right** qui permettent de ne stopper que la flottaison à droite ou à gauche, mais leur usage est beaucoup plus minoritaire que **clear: both**.

Nous allons faire quelques expériences avec la propriété **float** sans mettre fin à la flottaison dans un premier temps, puis nous regarderons ce que provoque l'usage de **clear: both**.

Cas pratique 1 : utilisation de float: left;

Fichier HTML

```html
<div class="container">
    <p class="flottante_jaune">
        Ce paragraphe va sortir du flux
    </p>
    <p class="orange">
        Ce paragraphe va se placer à droite du paragraphe
        précédent…
    </p>
</div>
```

Fig. 1 Cas 1 : Fichier HTML © Skill and You

Fichier CSS

```css
.flottante_jaune {
    float:left;
    margin:0;
    padding:10px;
    background-color:#ffff00;
    font-weight:bold
}

.orange {
    height:300px;
    padding:10px;
    background-color:#ffa500;
    font-weight:bold
}
```

Fig. 2 Cas 1 : Fichier CSS © Skill and You

Résultat dans un navigateur

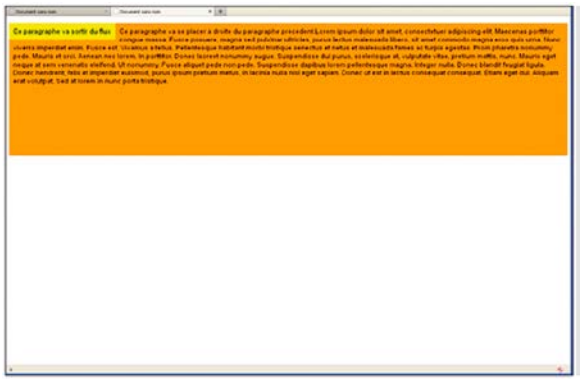

Fig.3 Cas 1 : Résultat dans le navigateur © Skill and You

Le navigateur place tout d'abord la boîte jaune, alignée sur le bord gauche de cette partie de notre page, puis dans l'espace laissé libre à sa droite, il affiche la boîte orange. Le flux occupe tout l'espace disponible : la boîte orange reprend donc toute la largeur de la page sitôt passée la limite inférieure de la boîte flottante jaune.

Cas pratique 2 : utilisation de float: right;

Fichier HTML

```
<div class="container">
    <p class="flottante_jaune">
        Ce paragraphe va sortir du flux
    </p>
    <p class="orange">
        Ce paragraphe va se placer à gauche du paragraphe
        précédent…
    </p>
</div>
```

Fig.4 Cas 2 : Fichier HTML © Skill and You

Fichier CSS

```
.flottante_jaune {
    float:right;
    margin:0;
    padding:10px;
    background-color:#ffff00;
    font-weight:bold
}

.orange {
    height:300px;
    padding:10px;
    background-color:#ffa500;
    font-weight:bold
}
```

Fig.5 Cas 2 : Fichier CSS © Skill and You

Résultat dans un navigateur

Fig.6 Cas 2 : Résultat dans un navigateur © Skill and You

Même chose que dans le premier exemple, mis à part que le flottant se place sur la droite.

Cas pratique 3 : utilisation de clear: both

Reprenons l'exemple 1 et appliquons la règle CSS clear à la boîte orange.

Fichier HTML

```
<div class="container">
    <p class="flottante_jaune">
        Ce paragraphe va sortir du flux
    </p>
    <p class="orange">
        Ce paragraphe va se placer à droite du paragraphe
        précédent…
    </p>
</div>
```

Fig.7 Cas 3 : Fichier HTML © Skill and You

Fichier CSS

```
.flottante_jaune {
    float:left;
    margin:0;
    padding:10px;
    background-color:#ffff00;
    font-weight:bold
}

.orange {
    clear:both;
    height:300px;
    padding:10px;
    background-color:#ffa500;
    font-weight:bold
}
```

Fig.8 Cas 3 : Fichier CSS © Skill and You

Résultat dans un navigateur

Fig. 9 Cas 3 : Résultat dans le navigateur © Skill and You

La boîte orange retrouve son flux normal et se place sous la boîte jaune.

Dans la pratique, si d'aventure nous nous trouvions en présence d'exemples assimilables aux deux premiers, c'est que le flottement aurait été mal réalisé car il y a risque de débordement (cf. leçon sur les débordements et le choix du positionnement). Le troisième exemple est envisageable et la réalisation correcte, néanmoins les trois principales utilisations des flottements ont trait **aux images, au colonnage et aux vignettes**. Nous allons étudier successivement chacune de ces trois utilisations.

II. Mise en pratique

A. Texte autour d'une image

Lorsque la propriété **float** voit le jour dans la première version du CSS, l'exemple donné dans les spécifications du W3C concerne une image. « Par exemple, en indiquant "left" comme la valeur correspondant à la propriété float pour une image, celle-ci est déplacée vers la gauche jusqu'à ce qu'elle atteigne la marge, le padding ou la bordure d'un autre élément de bloc » (Source : https://www.w3.org/TR/CSS1/#floating-elements). Et c'est justement pour les images, en grande partie, que les flottements ont été créés. Mais pourquoi ?

Regardons ce qu'il se passe lorsqu'on insère une image dans un paragraphe de texte.

Fichier HTML

```
<div class="container">
    <p>
        Vivamus sed nulla in libero venenatis blandit! Praesent
        pulvinar a ex ac commodo. Sed vitae lacus nec arcu
        mollis accumsan? <img src="img/oiseau.png" width="150"
        height="100" alt="oiseau">Integer dapibus turpis eget
        lacinia tristique. Nulla venenatis, elit id dictum
        maximus. Curabitur nec ultrices lacus. Vivamus viverra
        ullamcorper eros non sodales. Mauris cursus magna
        libero, vitae hendrerit urna fermentum id. Ut
        consectetur efficitur enim sed maximus. Curabitur nec
        ultrices lacus. Vivamus viverra ullamcorper eros non
        sodales. Mauris cursus magna libero, vitae hendrerit
        urna fermentum id. Ut consectetur efficitur enim sed
        maximus. Integer dictum rhoncus sagittis. Nullam semper
        dui non ligula mattis, nec blandit erat convallis! Etiam
        et eros tellus.Curabitur nec ultrices lacus. Vivamus
        viverra ullamcorper eros non sodales. Mauris cursus
        magna libero, vitae hendrerit urna fermentum id. Ut
        consectetur efficitur enim sed maximus.
    </p>
</div>
```

Fig. 10 Fichier HTML © Skill and You

Voyons maintenant ce qu'il se produit lorsqu'on fait flotter l'image sur la gauche.

Fichier HTML

```
<div class="container">
    <p>
        Vivamus sed nulla in libero venenatis blandit! Praesent
        pulvinar a ex ac commodo. Sed vitae lacus nec arcu
        mollis accumsan? <img src="img/oiseau.png" width="150"
        height="100" class="colonne photo-marge" alt="oiseau">
        Integer dapibus turpis eget lacinia tristique. Nulla
        venenatis, elit id dictum maximus. Curabitur nec
        ultrices lacus. Vivamus viverra ullamcorper eros non
        sodales. Mauris cursus magna libero, vitae hendrerit
        urna fermentum id. Ut consectetur efficitur enim sed
        maximus. Curabitur nec ultrices lacus. Vivamus viverra
        ullamcorper eros non sodales. Mauris cursus magna
        libero, vitae hendrerit urna fermentum id. Ut
        consectetur efficitur enim sed maximus. Integer dictum
        rhoncus sagittis. Nullam semper dui non ligula mattis,
        nec blandit erat convallis! Etiam et eros
        tellus.Curabitur nec ultrices lacus. Vivamus viverra
        ullamcorper eros non sodales. Mauris cursus magna
        libero, vitae hendrerit urna fermentum id. Ut
        consectetur efficitur enim sed maximus.
    </p>
    <div class="clear"></div>
</div>
```

Fig. 11 Fichier HTML © Skill and You

Application des classes

Les deux classes appliquées au même élément : « colonne » « photo_marge »

On observe également l'élément div.clear pour arrêter la flottaison.

Fichier CSS

```
.colonne {
    float:left;
}
.photo-marge {
    margin:10px 10px 10px 0;
}
.clear {
    clear:both;
}
```

Fig. 12 Fichier CSS © Skill and You

Résultat dans un navigateur

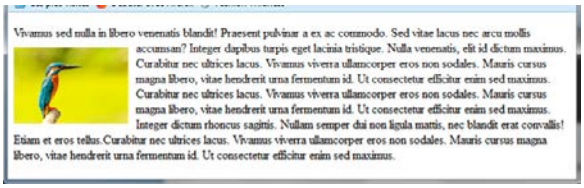

Fig. 13 Résultat dans un navigateur © Skill and You

Voilà qui est tout de même plus harmonieux.

B. Colonnage

Observons maintenant la manière de réaliser un colonage.

Fichier HTML

```
<div class="container">
    <div id="left" class="colonne w250">
        <h2> première colonne</h2>
    </div>
    <div id="center" class="colonne w450">
        <h2> deuxième colonne</h2>
    </div>
    <div id="right" class="colonne w250">
        <h2> troisième colonne</h2>
    </div>
    <div class="clear"></div>
</div>
```

Fig. 14 Fichier HTML © Skill and You

La dernière division clear
Ne l'oubliez surtout pas, elle permet aux objets HTML suivant de retrouver leur flux normal.

Fichier CSS

```
.container {
    margin:0 auto;
    width:1070px;
}
.colonne {
    float:left
    }
.clear {
    clear:both;
}
.w250 {
    width:250px;
    min-height:500px;
    margin:0 20px;
}
.w450 {
    width:450px;
    min-height:500px;
    margin:0 20px;
}
.w250, .w450{
    background-color:#ffff00;
}
h2 {
    text-align:center;
}
```

Fig. 15 Fichier CSS © Skill and You

Résultat dans un navigateur

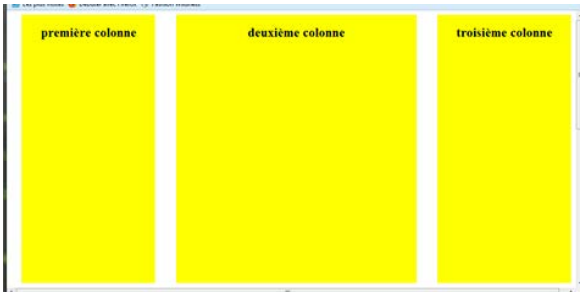

Fig. 16 Résultat dans un navigateur © Skill and You

Remarquez que la largeur des trois colonnes ne doit pas excéder la largeur du parent **div.container**. Je vous invite à modifier par vous-même différentes valeurs :

1. Augmenter les marges extérieures des colonnes

2. Réduire la largeur de **div.container** (après avoir rétabli les marges)

3. Ajouter un **padding** aux colonnes ou à div.container (après avoir rétabli la largeur de div.container)

4. Utiliser la règle **box-sizing: border-box** sur les éléments flottants et refaire les points 1, 2 et 3

C. Vignettes

Les vignettes sont un autre cas fréquent d'utilisation des flottements. On peut procéder comme suit :

Fichier HTML

```
<div class="container">
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="clear"></div>
</div>
```

Fig. 17 Fichier HTML © Skill and You

Fichier CSS

```css
.vignette {
    float:left;
    width:200px;
    height:200px;
    margin:20px;
    background-color:black;
}
.clear {
    clear:both;
}
```

Fig. 18 Fichier CSS © Skill and You

Résultat dans un navigateur

Fig. 19 Résultat dans un navigateur © Skill and You

Les **div.vignette** se placent les unes à côté des autres. Lorsqu'elles excèdent la largeur de **div.container,** elles continuent de se placer de la même manière sur une seconde rangée. Je vous invite à étudier le comportement des **div.vignette** lorsque l'on réduit la largeur de la fenêtre du navigateur. Vous comprendrez, dès lors, pourquoi les flottements ont connu un regain d'intérêt avec l'apparition du *Responsive Design* lorsqu'il s'est agi de travailler pour différentes tailles d'écran.

D. Méthodes alternatives pour stopper la flottaison

Nous allons voir très rapidement deux méthodes alternatives pour stopper la flottaison. La première nous permettra de rencontrer une nouvelle manière de cibler en CSS, la seconde a trait aux particularités que l'on nomme le « **contexte de formatage** ». Sachez toutefois que l'une ou l'autre demande un peu de pratique pour savoir où les placer exactement, c'est pourquoi, avant de les utiliser, il faut travailler d'abord avec un élément HTML sur lequel on ajoute une classe **.clear** comme nous l'avons fait précédemment.

Méthode dite du clearfix: after

Le pseudo-élément **after** permetde cibler un élément sans avoir à ajouter du HTML, on crée en quelque sorte un élément HTML virtuel que le CSS va affecter. Nous placions jusqu'alors notre **div.clear** après le dernier élément flottant ou après le paragraphe suivant l'image flottante. Ici, c'est sur le parent que nous allons travailler.

Fichier HTML

```html
<div id="wrapper" class="clearfix">
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
</div>
```

Fig. 20 Fichier HTML © Skill and You

Notez la présence de la classe **clearfix** sur le parent des éléments flottants.

Fichier CSS

```css
.vignette {
    float:left;
    width:200px;
    height:200px;
    margin:20px;
    background-color:black;
}
.clearfix:after {
    content:"";
    clear:both;
    display:block;
}
```

Fig. 21 Fichier CSS © Skill and You

Explication de clearfix: after :

content : permet de créer un élément html par le css, ici on doit indiquer une valeur entre guillemets, si l'on n'indique rien entre les guillemets on ne verra rien (c'est notre cas, nous voulons seulement arrêter la flottaison).

clear : comme pour un **div.clear**

display: block, il s'agit de créer un contenu de block (on peut également utiliser **table**).

On peut également ajouter une hauteur et une largeur nulle (**height: 0 ; width: 0**) et/ou **visibility: hidden** pour les plus prudents.

Méthode overflow

Nous venons de voir la méthode avec le pseudo-élément, celle que nous allons aborder est encore plus simple à réaliser et, comme la précédente, se place sur le parent.

Le HTML ne change pas (sauf le nom de la classe qui stoppe la flottaison).

```
<div id="wrapper" class="clearfix-overflow">
    <div class="vignette"></div>
    <div class="vignette"></div>
    <div class="vignette"></div>
</div>
```

Fig. 22 Fichier HTML © Skill and You

Pour le CSS, on aura :

```
.vignette {
    float:left;
    width:200px;
    height:200px;
    margin:20px;
    background-color:black;
}
.clearfix-overflow {
    overflow:hidden;
}
```

Fig. 23 Fichier CSS © Skill and You

La règle **overflow: hidden** indique que les éléments qui dépassent de **div#wrapper** seront invisibles, ce faisant il modifie le comportement habituel d'un élément de block. Le contexte de formatage de **div#wrapper** est modifié, le forçant à prendre en compte les tailles de tous ses enfants y compris les flottants.

Dans un premier temps, comme dit plus haut, il vaut mieux assimiler correctement la réalisation des flottements et de la fin de flottaison avec la méthode **div. clear** avant, petit à petit, de passer aux deux autres méthodes décrites dans cette dernière partie du cours.

Il est primordial de connaître les flottements pour l'élaboration de pages HTML, car c'est encore de nos jours une méthode de mise en page très utilisées qui, de plus, est prise en compte même par les vieux navigateurs.