Méthode position

Dans un précédent chapitre, nous avons identifié quatre solutions pour réaliser des mises en page, à savoir : **flexbox**, **flottements**, méthode **display** et méthode **position**. Dans ce chapitre, il s'agira d'élargir notre gamme en s'intéressant à la règle position. Elle a la particularité, contrairement aux précédentes, d'être utilisé pour des cas très particuliers qui, très souvent, ne peuvent être réalisés au moyen d'autres méthodes.
Le principe fondamental à retenir est qu'un élément utilisant la méthode **position** doit avoir un repère par rapport auquel il va être placé et/ou déplacé. En position relative, le repère est le positionnement naturel de l'élément ; en position absolue, c'est le premier parent positionné ; en position fixe, c'est l'écran.

I. Position par défaut (rappel)

Comme nous l'avons vu précédemment, la position par défaut des éléments de bloc se trouve être sur le coin supérieur gauche de son parent.

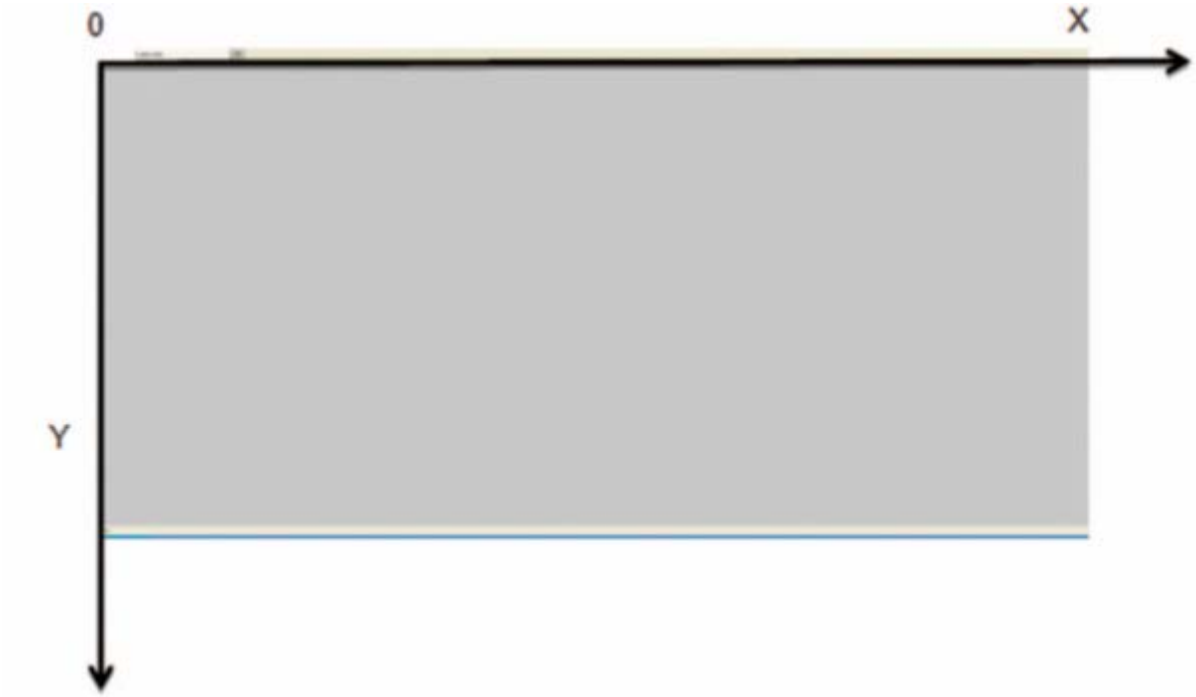

Fig. 1 *Pos1 © DR*

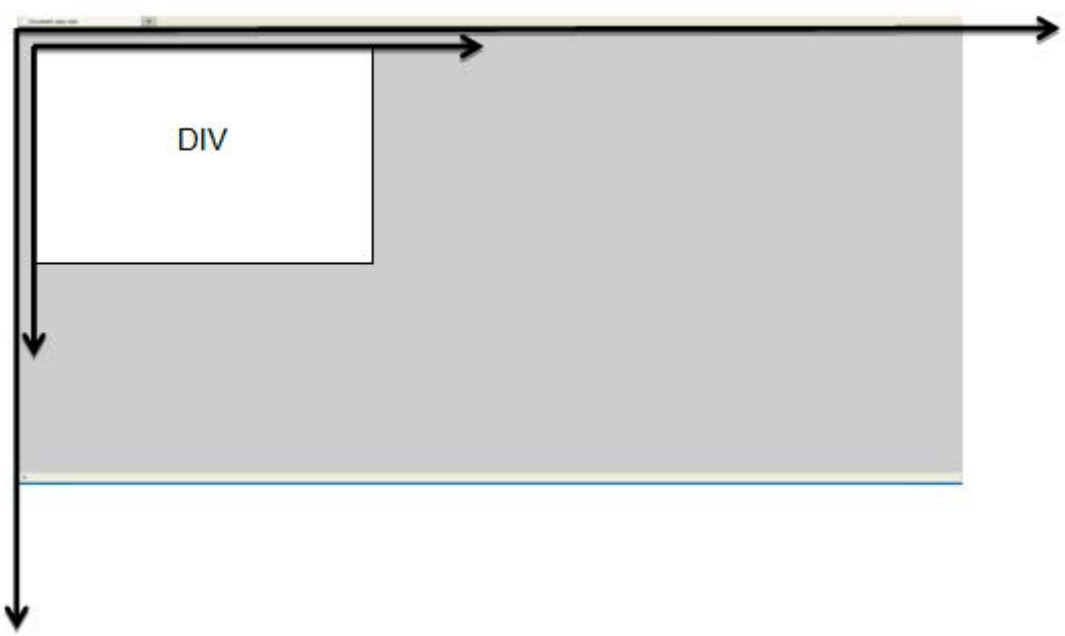

Fig. 2 Pos2 © DR

II. Position relative

En CSS, la position relative permet de sortir un élément du flux normal – mais pas complètement, comme nous le verrons plus précisément dans l'exemple 3. L'élément en position relative est déplacé de l'endroit où il devrait apparaître par défaut.

Syntaxe CSS :

```
position:relative;
top:5px;
left:-10px;
```

Fig. 3 Pos3 © DR

Ici, l'élément ayant ces règles CSS se déplace (par rapport à sa position naturelle) de 5px vers le bas et 10px vers la gauche (car la valeur est négative).

Explorons cette méthode grâce à quelques exemples.

Exemple 1
Fichier HTML

```
<body>

    <div id="conteneur_rouge">ici le contenu 1</div>

</body>
```

Fig. 4 Pos4 © DR

Fichier CSS

```css
#conteneur_rouge {
    width:500px;
    height:300px;
    position:relative;
    top:150px;
    left:300px;
    background-color:red;
}
```

Fig.5 Pos5 © DR

Résultat dans un navigateur

Fig.6 Pos6 © DR

La règle CSS top est la distance suivant l'axe des ordonnées (Y) et la règle left la distance suivant l'axe des abscisses (X).

Exemple 2
Fichier HTML

```html
<div id="conteneur_rouge">
    <div id="contenu_orange">ici le contenu 1</div>
</div>
```

Fig.7 Pos7 © DR

Fichier CSS

```css
#conteneur_rouge {
    width:500px;
    height:300px;
    position:relative;
    top:150px;
    left:300px;
    background-color:red;
}

#contenu_orange {
    width:200px;
    height:200px;
    position:relative;
    top:-50px;
    left:-100px;
    background-color:orange;
}
```

Fig.8 Pos8 © DR

Résultat dans un navigateur

Fig.9 Pos9 © DR

Comme vous pouvez le remarquer, nous pouvons positionner les div avec des valeurs négatives.

Ces derniers se positionnent par rapport à **leur position naturelle dans le flux normal**, donc la position des éléments dans **le code HTML** est très importante.

Exemple 3

Nous rajoutons une div qui reste dans le flux (celle qui se trouve en vert) juste après une div qui est sortie du flux.

Fichier HTML

```
<div id="conteneur_rouge">
    <div id="contenu_orange">ici le contenu 1</div>
    <div id="conteneur_vert">ici le contenu 2</div>
</div>
```

Fig. 10 Pos10 © DR

Fichier CSS

```
#conteneur_vert {
    width:200px;
    height:200px;
    background-color:green;
}
```

Fig. 11 Pos11 © DR

(On indique uniquement le nouvel élément)

Résultat dans un navigateur

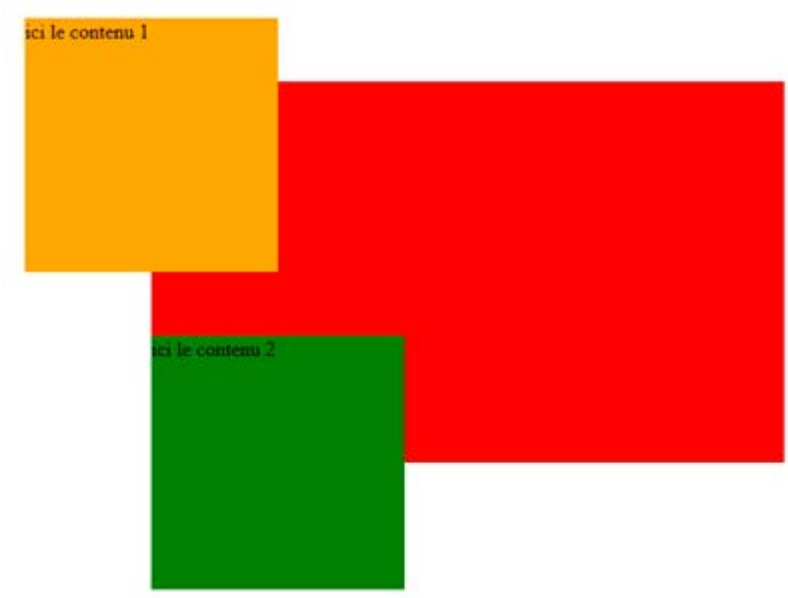

Fig. 12 Pos12 © DR

Nous remarquons que la **div contenu_2** se positionne comme si la **div contenu_1** était restée dans le flux normal.

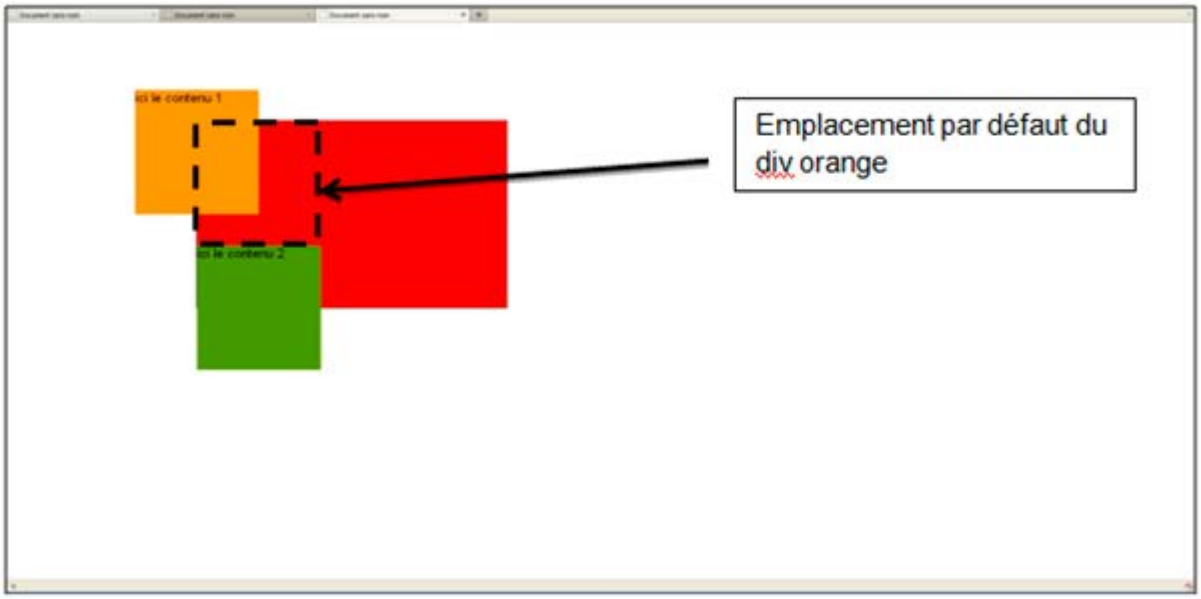

Fig.13 Pos13 © DR

Cet exemple vous aidera certainement à mieux maîtriser la position relative d'un élément de bloc. Retenez qu'outre le déplacement par rapport à son positionnement naturel, la place qu'il devrait occuper dans le flux normal est comme réservée.

III. Mise en pratique de la position relative

Voyons comment appliquer concrètement ce que l'on a appris sur la position relative. Nous allons prendre l'exemple d'un élément que l'on veut légèrement déporter de son flux naturel afin de créer un effet de décalage.

Le fichier HTML

```
<div id="wrapper">
    <header>
        <h1>Titre de la page</h1>
    </header>
    <main class="main">
        <article class="article">
            <h2>Titre de l'article</h2>
            <p>
                Fusce ipsum lectus, ultrices quis efficitur nec, lacinia
                vitae elit. Ut feugiat fringilla gravida. Proin consequat,
                ex quis hendrerit molestie, magna orci consequat lectus, sed
                aliquet purus nisi nec arcu. Mauris bibendum quis lorem sed
                interdum. Vivamus tincidunt, massa at pretium rhoncus, mi
                lacus ultrices sem, ut gravida sapien metus vitae nulla.
                Fusce nulla justo, volutpat at faucibus scelerisque,
                condimentum quis quam. Duis lacinia consectetur vestibulum.
            </p>
        </article>
        <article class="article">
            <h2>Titre de l'article</h2>
            <p>
                Fusce ipsum lectus, ultrices quis efficitur nec, lacinia
                vitae elit. Ut feugiat fringilla gravida. Proin consequat,
                ex quis hendrerit molestie, magna orci consequat lectus, sed
                aliquet purus nisi nec arcu. Mauris bibendum quis lorem sed
                interdum. Vivamus tincidunt, massa at pretium rhoncus, mi
                lacus ultrices sem, ut gravida sapien metus vitae nulla.
            </p>
        </article>
    </main>
    <footer>
        <p>le footer et le copyright</p>
    </footer>
</div>
```

Fig. 14 Pos14 © DR

On a stylisé le site de manière à obtenir le résultat suivant :

Fig. 15 Pos15 © DR

Pour rendre la mise en page plus dynamique, nous souhaitons décaler le cartouche du titre des articles sur la gauche afin qu'il déborde légèrement du conteneur. Nous allons donc travailler sur la balise **h2** qui a pour l'instant le code succinct suivant :

```css
h2 {
    margin:25px 0;
    background: #ff9933;
    color: #fff;
    padding:10px;
    width: 40%;
    font-family: arial;
}
```

Fig. 16 Pos16 © DR

On indique d'abord le mode de positionnement :

```css
position: relative;
```

Fig. 17 Pos17 © DR

On décale notre bloc titre de 60px vers la gauche :

```css
left:-60px;
```

Fig. 18 Pos18 © DR

Enfin, on compense ce décalage en augmentant légèrement le **padding-left** du **h2**, afin que le texte du titre ne sorte pas du **#wrapper**.

```
padding-left: 25px;
```

Fig. 19 Pos19 © DR

Le code CSS de **h2** devient :

```
h2 {
    margin:25px 0;
    background: #ff9933;
    color: #fff;
    width: 40%;
    font-family: arial;
    position: relative;
    left:-60px;
    padding:10px 10px 10px 25px;
}
```

Fig. 20 Pos20 © DR

Et notre rendu :

Fig. 21 Pos21 © DR

Notez que l'on aurait pu arriver au même résultat avec un **margin-left** négatif. Quelle différence entre les deux ? Comme nous l'avons vu plus haut, dans le positionnement relatif l'élément déplacé conserve tout de même l'espace qu'il aurait occupé dans le flux naturel. Ce qui n'aurait pas été le cas d'une marge négative. Au passage, l'emploi de marges négatives surtout à des degrés importants (-250px, -100px, etc.) est dans la plupart des cas le signe d'un problème de choix ou de non-choix de positionnement. Les marges ne doivent pas se substituer aux méthodes de positionnement et de mise en page que nous avons vues auparavant.

IV. Position absolue

La position absolue d'un élément de bloc est similaire à la position relative, à la différence que :

– l'élément sort complètement du flux normal, ce qui laisse une plus grande marge de manœuvre pour le positionner dans une page ;

– sa position dans le fichier HTML est moins importante que pour un élément en position relative. En effet, l'élément en position absolue se positionne par rapport à son premier parent lui-même positionné (en absolu ou relatif). Et si aucun parent n'est positionné, c'est le système de coordonnées globales de la page qui sert de référence. Dans ce cas, sa position dans le fichier HTML n'a plus aucune importance ;

– enfin, la particularité d'un élément en position absolue, est qu'il possède un troisième axe local (Z) dit de profondeur. C'est-à-dire que l'on peut choisir entre deux éléments en position absolue celui qui se place au-dessus de l'autre en cas de chevauchement.

Ce nouveau concept est peut-être encore flou pour vous, mais nous allons le développer avec des exemples concrets.

Syntaxe CSS

Tableau n°1 Syntaxe CSS du positionnement

position;absolute;top:5px;left:10px;	Le positionnement se fait par rapport à la bordure supérieure et gauche de son premier parent positionné
position;absolute;top:5px;right:10px;	Le positionnement se fait par rapport à la bordure supérieure et droite de son premier parent positionné
position;absolute;bottom:5px;left:10px;	Le positionnement se fait par rapport à la bordure inférieure et gauche de son premier parent positionné
position;absolute;bottom:5px;right:10px;	Le positionnement se fait par rapport à la bordure inférieure et droite son de premier parent positionné

Exemple 1
Fichier HTML

```
<body>

    <div id="conteneur_rouge">ici le contenu 1</div>

</body>
```

Fig. 22 Pos22 © DR

Fichier CSS

```css
#conteneur_rouge {
    width:500px;
    height:300px;
    position: absolute;
    top:5%;
    right:20px;
    background-color:red;
}
```

Fig. 23 Pos23 © DR

Résultat dans un navigateur

Fig. 24 Pos24 © DR

Explication : le seul parent de conteneur_rouge est **<body>**. Par défaut, **<body>** est positionné, c'est donc le repère du positionnement demandé, conteneur_rouge se positionne par rapport à **<body>** à 5 % du haut et à 20px de la droite.

Exemple 2

Cet exemple est similaire à l'exemple 3 du chapitre position relative. Observez la différence de comportement.

Fichier HTML

```html
<div id="conteneur_rouge">
    <div id="conteneur_orange"></div>
    <div id="conteneur_vert"></div>
</div>
```

Fig. 25 Pos25 © DR

Fichier CSS

```css
#conteneur_rouge {
    width:500px;
    height:300px;
    margin-top:150px;
    /* j'utilise les marges pour démontrer que la position
    de ce bloc n'a pas d'importance */
    margin-left:300px;
    background-color:red;
    position:relatif;
    /* ou position:absolute, même résulat;
    le fait de le mettre en relatif est une convention
    pour signifier que cet élément va servir de premier parent positionné
    pour les deux blocs qui suivent.
    On indique donc ni top ni bottom ni rien.
    L'élément est en relatif, sans indication de position,
    donc il se trouve en fait dans la position naturelle du flux. */
}

#conteneur_orange {
    width:200px;
    height:200px;
    position: absolute;
    top:-50px;
    left:-50px;
    background-color:orange;
}

#conteneur_vert {
    width:200px;
    height:200px;
    background-color:green;
}
```

Fig. 26 Pos26 © DR

Résultat dans un navigateur

Fig. 27 Pos27 © DR

Nous remarquons que **div.orange** se positionne par rapport à son parent qui est en position relative.

Mais vous remarquez aussi que **div.vert** (flux normal) se place comme si **div.orange** n'existait pas, ce qui n'était pas le cas pour la position relative. On assiste donc à une superposition des éléments sur plusieurs couches (ou **layers**).

Nous vous conseillons maintenant de tester ce code HTML dans lequel nous avons inversé 2 lignes, avec le même code CSS.

Fichier HTML

```
<div id="conteneur_rouge">
    <div id="conteneur_orange"></div>
    <div id="conteneur_vert"></div>
</div>
```

Fig. 28 Pos28 © DR

Vous remarquerez que le résultat est identique à la première version.

En conséquence, l'emplacement d'un élément en position absolue dans le code HTML n'a aucune importance à partir du moment où son parent direct est positionné. Sa position dans la page sera fonction de la position de son parent en position relative (ou absolue).

***Exemple 3** : superposition des div*

Nous pouvons donc superposer des éléments de bloc et choisir l'ordre d'empilage. La règle de style qui permet cela est :

z-index:(ici un nombre supérieur ou égale à 1)

L'élément ayant un z-index de 2 recouvre celui qui possède un z-index de 1.

Fichier HTML

```
<body>

    <div id="conteneur_orange">ici le contenu 1</div>
    <div id="conteneur_vert">ici le contenu 2</div>

</body>
```

Fig. 29 Pos29 © DR

Fichier CSS

```css
#conteneur_orange {
    width:200px;
    height:200px;
    position: absolute;
    top:150px;
    left:350px;
    background-color:orange;
    z-index:2;
}

#conteneur_vert {
    width:200px;
    height:200px;
    position: absolute;
    top:250px;
    left:450px;
    background-color:green;
    z-index:1;
}
```

Fig.30 Pos30 © DR

Fig.31 Pos31 © DR

Si on inverse les z-index, on aura le code suivant :

```
#conteneur_orange {
    width:200px;
    height:200px;
    position: absolute;
    top:150px;
    left:350px;
    background-color:orange;
    z-index:1;
}

#conteneur_vert {
    width:200px;
    height:200px;
    position: absolute;
    top:250px;
    left:450px;
    background-color:green;
    z-index:2;
}
```

Fig.32 Pos32 © DR

Le rendu est donc :

Fig.33 Pos33 © DR

Superposition des div
Si aucun z-index n'est précisé, c'est la position des div dans le code HTML qui prévaut pour la superposition. C'est-à-dire que la deuxième div recouvre la première.

Donc, comme le conteneur vert arrive après le conteneur orange, le code précédent est équivalent au code suivant sans z-index :

```
#conteneur_orange {
    width:200px;
    height:200px;
    position: absolute;
    top:150px;
    left:350px;
    background-color:orange;
}

#conteneur_vert {
    width:200px;
    height:200px;
    position: absolute;
    top:250px;
    left:450px;
    background-color:green;
}
```

Fig. 34 Pos34 © DR

Donnera :

Fig. 35 Pos35 © DR

L'utilisation de **z-index** pour indiquer l'emplacement des couches n'est donc pas toujours nécessaire et dépend de la structuration du HTML.

V. Mise en pratique de la position absolue

L'une des utilisations les plus courantes de la position absolue consiste à réaliser des superpositions d'éléments avec apparition-disparition.

Exemple, une galerie où l'image apparaît au rollover :

Fig. 36 Pos36 © DR

Le HTML des quatre vignettes est le suivant :

```
<figure>
    <a href="#">
        <img src="img/01.jpg" >
        <figcaption>Morbi tincidunt tortor non vehicula eleifend.</figcaption>
    </a>
</figure>
```

Fig. 37 Pos37 © DR

Il est répété 4 fois. Nos images font 150px sur 150px, on va donc styliser le bloc **figure** pour qu'il fasse cette largeur.

Nous avons utilisé deux balises HTML encore inconnues : **<figure>** et **<figcaption>**. **<figure>** définit une zone de contenu cohérente comprenant une ou plusieurs images ou schémas. On peut ajouter une légende à l'élément illustratif avec la balise **<figcaption>**.

Voyons le CSS de la zone d'illustration :

```
figure {
    border: 1px solid #ddd;
    float: left;
    width: 150px;
}
```

Fig. 38 Pos38 © DR

Pour le cartouche sur l'image, nous allons donner une taille, graisser la police, ajouter un petit padding et surtout donner un fond de couleur sombre.

```css
figcaption {
    font-size: 18px;
    font-weight: bold;
    padding: 20px 10px;
    background: #444;
}
```

Fig. 39 Pos39 © DR

N'oublions pas de donner une couleur à notre lien (**<a>** encadre **<figcaption>**) qui soit lisible sur un fond noir et de supprimer le soulignement :

```css
a{
    color: #fff;
    text-decoration: none;
}
```

Fig. 40 Pos40 © DR

On obtient le résultat suivant :

Fig. 41 Pos41 © DR

Maintenant, on veut placer le bloc de texte par-dessus l'image. Pour cela on va placer **figure** en relatif et **figcaption** en **absolute**. En effet, **img** se trouve au même point d'origine que **figure**. On va donc indiquer **top:0** et **left:0** pour qu'ils se superposent.

```
figure {
    border: 1px solid #ddd;
    float: left;
    width: 150px;
    position: relative;
}
figcaption {
    font-size: 18px;
    font-weight: bold;
    padding: 20px 10px;
    background: #444;
    position: absolute;
    top:0;
    left:0;
 }
```

Fig. 42 *Pos42 © DR*

Le résultat :

Fig. 43 *Pos43 © DR*

On s'approche, mais il reste une partie de l'image visible. Comment faire ? Indiquer à l'image et au cartouche une hauteur identique ? C'est possible, mais il faudrait soustraire à cette taille le **padding**. Le plus simple dans ce cas est de placer toutes les valeurs à zéro (top, left, bottom et top) afin que l'élément en position absolue recouvre toutes les parties du parent quelle que que soit sa taille. Soit :

```
figcaption {
    font-size: 18px;
    font-weight: bold;
    padding: 20px 10px;
    background: #444;
    position: absolute;
    top:0;
    left:0;
    right:0;
    bottom:0;
 }
```

Fig. 44 *Pos44 © DR*

On obtient le résultat suivant, ce qui est déjà beaucoup mieux.

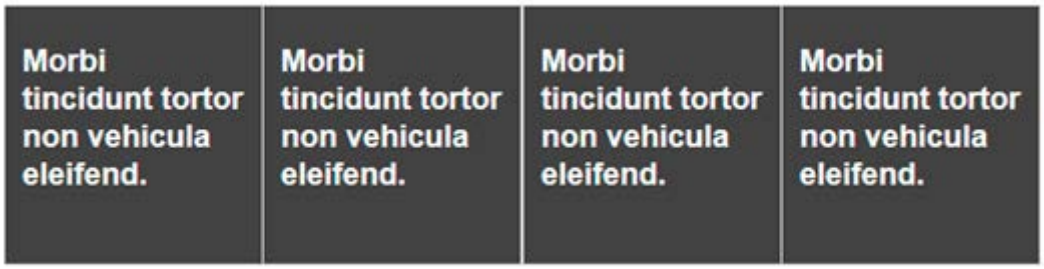

Fig.45 Pos45 © DR

Avant de réaliser l'apparition-disparition, on va modifier la couleur du fond du bloc pour laisser apparaître un peu de l'image de fond. Pour ce faire, on utilise une déclaration de couleur en RVBA :

```
figcaption {
    font-size: 18px;
    font-weight: bold;
    padding: 20px 10px;
    background: rgba(10,10,10,0.6);
    /*rgba = red, green, bleu, alpha la couche alpha gère la transparence*/
    position: absolute;
    top:0;
    left:0;
    right:0;
    bottom:0;
}
```

Fig.46 Pos46 © DR

Rendu :

Fig.47 Pos47 © DR

Maintenant, on va tout simplement faire disparaître le bloc de texte (avec la règle **display:none**) au passage de la souris. On pourrait penser de prime abord qu'on va le demander au survol du texte (avec **figcaption:hover**), mais en réalité il vaut mieux cibler le parent d'**img** et **figcaption** (faites un essai, vous comprendrez).

Comment faire ? Au passage de la souris sur figure (quand **figure:hover**) on modifie **figcaption**, soit :

```
figure:hover figcaption{
    display: none;
}
```

Fig. 48 Pos48 © DR

On aurait également pu mettre à profit la visibilité du bloc :

```
figure:hover figcaption{
    visibility: hidden;
}
```

Fig. 49 Pos49 © DR

Le rendu :

Fig. 50 Pos50 © DR

On remarque que le bloc de texte déborde un peu en bas, on va donc indiquer une hauteur à **figure** équivalente à celle de l'image.

```
figure {
    border: 1px solid #ddd;
    float: left;
    width: 150px;
    height: 150px;
    position: relative;
}
```

Fig. 51 Pos51 © DR

Et voilà !

VI. Position fixe

Comme dans un positionnement absolu, le contenu concerné est retiré totalement du flux. Mais il est cette fois positionné uniquement par rapport aux limites de la zone de visualisation, autrement dit la fenêtre du navigateur.

Le défilement de la page n'a aucun effet sur un contenu en position fixe.

Syntaxe CSS

Exemple *: cadre fixe en bas*

Fichier HTML

```
<div id="wrapper" >
        <h1 class="center"> Le flop du fixe </h1>
        <nav class="menu">
            <ul >
                <li>lien_1</li>
                <li>lien_2</li>
                <li>lien_3</li>
            </ul>
        </nav>
</div>
```

Fig.52 Pos53 © DR

Fichier CSS

```css
#wrapper {
    margin: 0px auto;
    /*50px pour dégager le titre de la page*/
    width: 70%;
    padding-top: 50px;
    height:300vh;
    /* présent uniquement pour avoir
    un défilement vertical de la page*/
    background-color:#7d38aa;
}
.menu {
    width:100%;
    height:50px;
    position:fixed;
    top:0;
    right:0;
    background-color:#385ca9;
}

ul, li {
    list-style: none;
}
ul {
   width:70%;
   /*même largeur que le #wrapper*/
   margin:0 auto;
}
li {
    display: inline-block;
    padding: 10px;
    color:#fff;
}
```

Fig. 53 Pos54 © DR

Résultat dans un navigateur

Fig. 54 Pos55 © DR

Pour placer le menu en bas de l'écran, il suffit d'indiquer :

```
.menu {
    width:100%;
    height:50px;
    position:fixed;
    bottom:0;
    /*on place le menu en bas*/
    right:0;
    background-color:#385ca9;
}
```

Fig. 55 Pos56 © DR

Le rendu (ne pas oublier de supprimer les 50px de marge haute sur le **#wrapper**) :

Fig. 56 Pos57 © DR

VII. Position collante (sticky)

La position dite collante est un mélange de position par défaut (**position: static**), c'est-à-dire que le bloc en position **sticky** se comporte comme un bloc normal pendant le scroll jusqu'au moment où il se comporte comme un bloc en position fixe.

Exemple
La colonne de gauche nommée **section.actualites** comporte de nombreux éléments qui obligent à faire défiler la page.

```
<main class="main">
    <section class="actualites">
        <article class="article top-article">
            <h2>title</h2>
            <p>qdfsq</p>
        </article>
        <!-- de nombreux articles sur la droite
        répétant cette matrice  -->
    </section>
    <aside class="column-right">
        <div class="add">
            <h2>Je colle </h2>
            <h3>Mais grave</h3>
            <p>Après le scroll, je bouge pas !</p>
        </div>
    </aside>
</main>
```

Fig. 57

Le CSS

```css
.main {
    display: flex;
    padding: 10px;
    min-height: 90vh;
}
.actualites {
    width: 70%;
    background-color: #ddd;
    min-height: 90vh;
}
.article {
    border-bottom: 2px dashed #fff;
    padding:55px 10px 30px 10px;
}
.top-article {
    background-color: #cceedd;
}
.column-right {
    width: 25%;
}
.add {
    padding: 20px;
    border: 1px solid #ddd;
    background: #fff;
}
```

Fig. 58

Méthode flexbox

Nous avons choisi la méthode flexbox pour mettre en forme les deux colonnes pour varier un peu, d'une part, et d'autre part afin d'être certain que div.column-right aura la même hauteur que la colonne de gauche et sera donc bien plus grand que div.add (nous en verrons la raison plus tard).

À ce stade, on obtient :

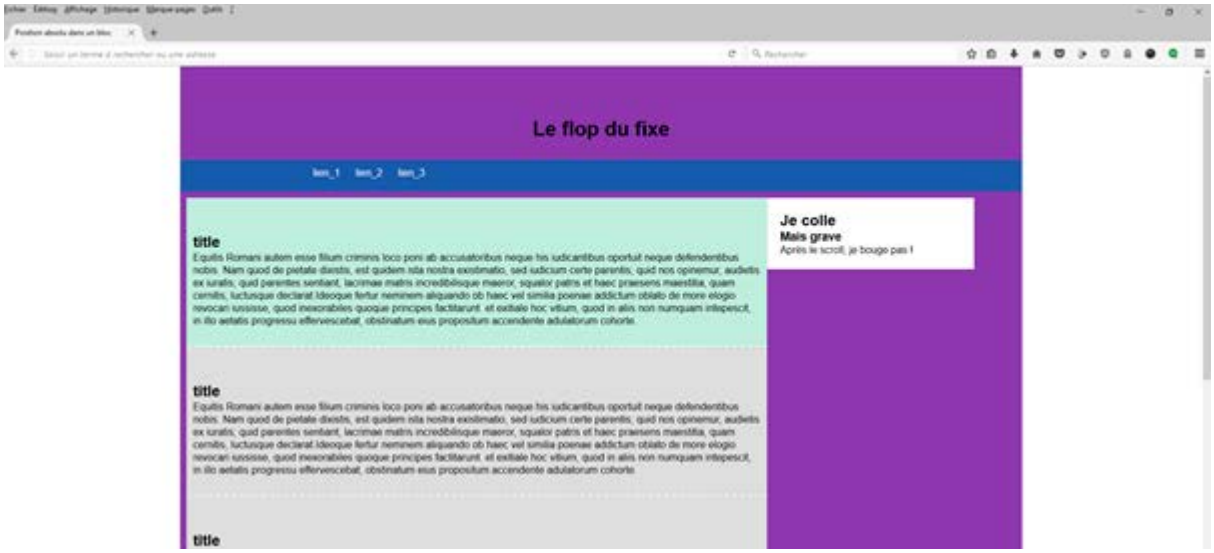

Fig. 59

Et si nous faisons défiler la page :

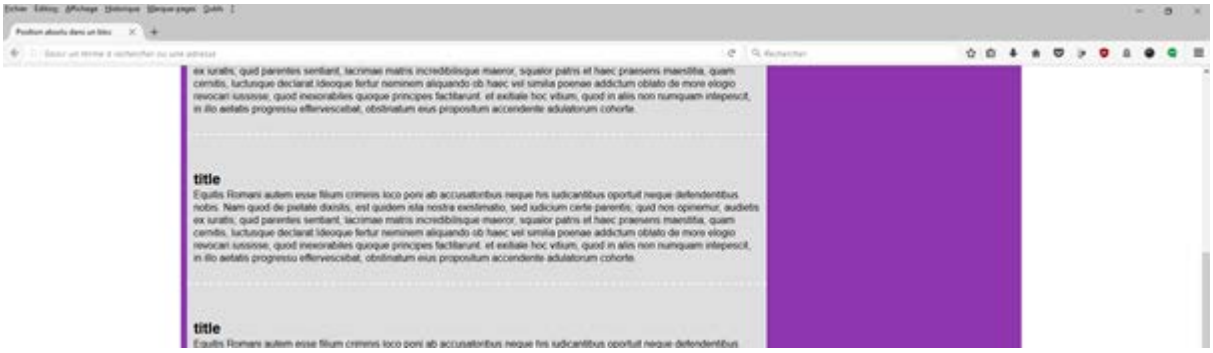

Fig. 60

Mais nous voulons que ce bloc reste en haut de page ? Quel est le secret du positionnement collant ? Il faut que l'élément collant soit plus petit que la colonne de gauche. Sinon, l'élément n'aura pas de place pour coulisser et remontera avec le reste du contenu. Ajoutons une règle en pixels en fond de la colonne de droite pour mieux visualiser tout cela :

```
.column-right {
    width: 25%;
    background: url(regle.jpg) top right repeat-y;
}
```

Fig. 61

Résultat

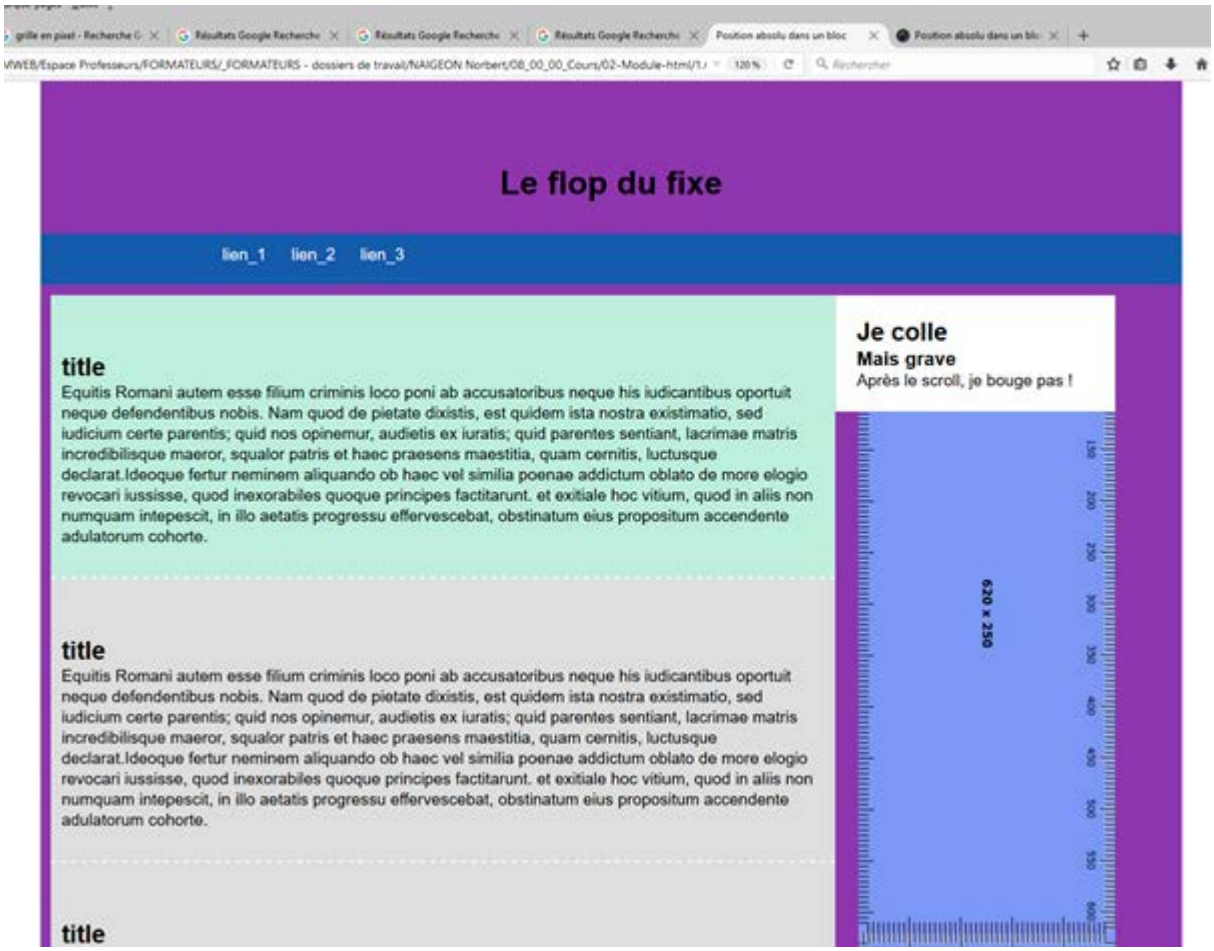

Fig. 62

L'élément collant va donc se déplacer le long de l'image numérotée, lorsque l'on scrolle, nous permettant d'observer le comportant collant qui est obtenu en indiquant sur **.add** :

```
.add {
    position: sticky;
    top:30px;
    padding: 20px;
    border: 1px solid #ddd;
    background: #fff;
}
```

Fig. 63

La première ligne demande à l'élément d'être collant (**position:sticky**) puis on choisit la position avec **top** (ici à 30px du haut de l'écran), ce qui nous donne au défilement :

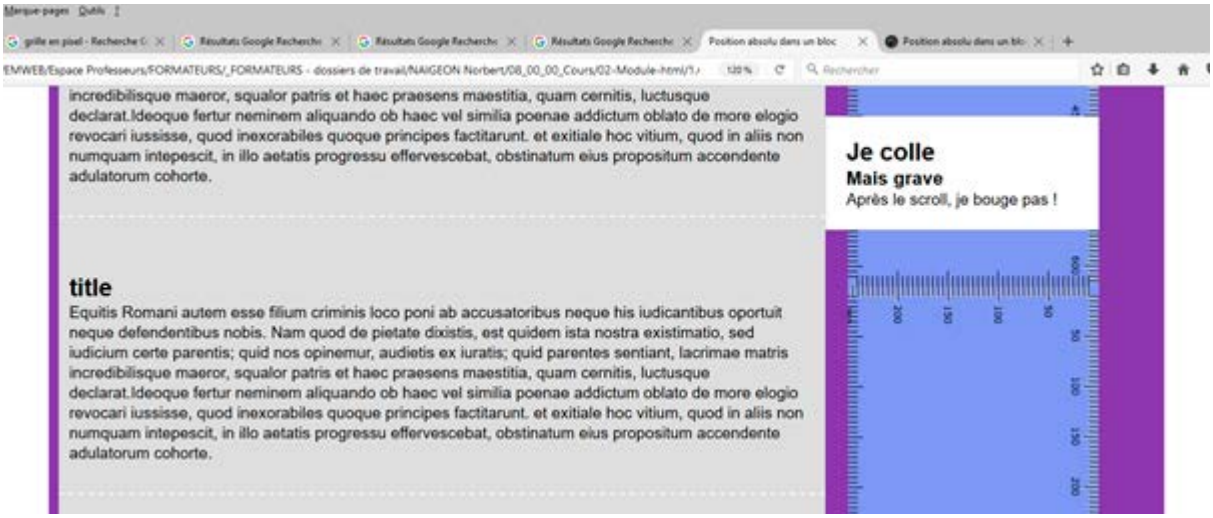

Fig. 64

Si on indique une hauteur de 350px à notre colonne de droite (avec un **padding-bottom** afin d'observer l'image de fond) :

```css
.column-right {
    width: 25%;
    background: url(regle.jpg) top right repeat-y;
    height: 350px;
    padding-bottom: 50px;
}
```

Fig. 65

Jusqu'à 350px, on a le même comportement :

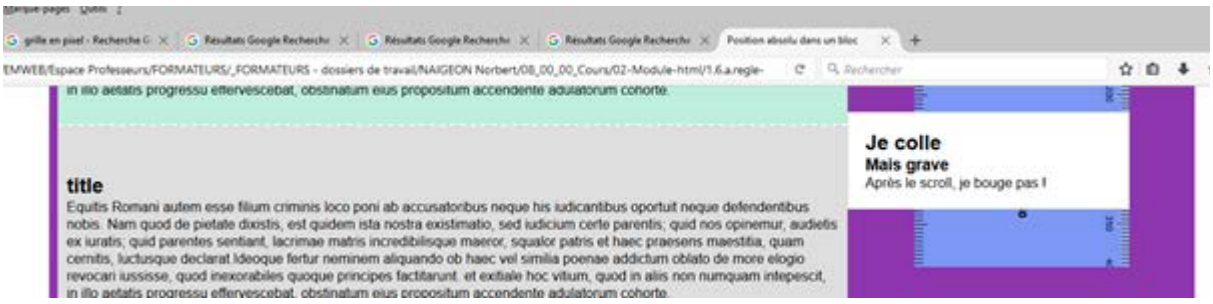

Fig. 66

Mais ensuite **div.add** remonte comme un contenu normal :

Fig. 67

Cette leçon vous aura permis de comprendre les différents comportements de la méthode **position**. Les éléments de blocs se superposent verticalement dans le flux normal. En positionnement relatif, le bloc se positionne par rapport à sa position naturelle dans le flux. En positionnement absolu, c'est en rapport avec son premier parent positionné qu'il le fait. Enfin, la position fixe est en rapport avec la zone de l'écran et la position collante est un mélange de positionnement par défaut et de position fixe.