# Débordements

Dans ce chapitre, nous allons regarder de plus près certaines configurations de choix de mise en page problématiques. En effet, il arrive qu'une méthode de sortie des éléments du flux naturelle aboutisse à des problèmes de rendus.
Le contenu d'un site web, en effet, ne doit pas seulement être envisagé dans le cadre d'une mise en page particulière où une image a une taille définitive et un bloc de texte possède un nombre de caractères précis (car même si on peut faire un choix de maquette à taille fixe cela reste rare). Non, il faut envisager les choses de manière « dynamique «. C'est-à-dire imaginer que nos éléments médias ou textes puissent ne pas occuper le même espace d'une page à l'autre. C'est ce travail d'imagination et d'anticipation qu'avec l'expérience vous ferez sans vous en rendre compte.
Nous allons pour chaque cas rappeler le comportement normal avant d'observer le comportement déviant, puis donner une solution. Enfin, nous essaierons de donner des critères de choix pour la réalisation de telle ou telle mise en page.

## I. Positionnement absolu et débordement

Observons le code suivant.

HTML :

```
<body>
    <div id="conteneur">
        <div id="jaune">bloc 1</div>
        <div id="vert">bloc 2</div>
    </div>
    <div id="bleu"> bloc 3</div>
</body>
```

*Fig. 1*

CSS :

```
* {
    margin:0px;
    padding:0px;
}

body {
    width:600px;
    font: 24px arial;
    color:000;
    font-weight:bold;
    margin:25px auto;
}

#conteneur{
    background-color:#a6a6a6;
    border:5px solid #333;
    position:relative;
}
```

*Fig. 2*

```
#jaune{
    background-color:#ccdd55;
    height:130px;
    width:400px
}

#vert {
    background-color:#55AA55;
    width:180px;
    height:50px;
    position:absolute;
    top:0px;
    right:0;
}

#bleu {
    height:50px;
    background-color:#00aadd;
    border:5px solid #3ee;
}
```

*Fig. 3*

Ce qui nous donne :


*Fig. 4*

Ici tout va bien, mais ce n'est que la théorie. Dans la vraie vie, on n'aura jamais de bloc_vert, bloc_jaune et tutti quanti, on aura des blocs dans lesquels viendront du texte, des images, des vidéos, entre autres contenus. Cette fois-ci, on va s'amuser à mettre des images et des textes.

Soit le HTML :

```html
<div id="conteneur">
    <div id="jaune">
        <p>
            Lorem ipsum dolor sit amet, consectetur adipiscing
            elit. Maecenas euismod viverra neque, et ornare
            sapien rhoncus eu. Interdum et malesuada fames ac
            ante ipsum primis in faucibus. Maecenas vel sem
            luctus, dapibus nulla in, tincidunt diam. Mauris
            eleifend dui vel congue tristique. Morbi auctor,
            tortor et malesuada hendrerit, purus libero pretium
            lacus, et malesuada ligula erat eu ante. Cras cursus
            eget libero id fermentum. Ut vel lobortis quam. Donec
            venenatis porttitor sem, quis egestas diam laoreet
            rutrum. Donec feugiat, sapien ut consectetur
            pellentesque, magna mi dignissim ipsum, id varius
            turpis nisi vel arcu. Proin malesuada lacus faucibus
            tellus auctor, ac commodo dolor bibendum.
        </p>
        <p>
            Integer quis malesuada nunc. Vivamus ut rutrum augue.
            Nulla sed tempus augue. Nunc aliquam libero porta
            euismod tempus. Nullam pulvinar felis iaculis dolor
            lacinia, ac aliquet augue iaculis. Nunc fringilla
            neque vel metus scelerisque suscipit. Donec feugiat
            adipiscing erat, at malesuada quam facilisis vel.
            Aliquam tempor risus at eros pellentesque ornare.
            Nunc congue diam sit amet facilisis suscipit. Sed
            fermentum lobortis adipiscing. Cum sociis natoque
            penatibus et magnis dis parturient montes, nascetur
            ridiculus mus. Sed imperdiet ut leo a auctor. Nam
            quis eleifend metus. Nam imperdiet sollicitudin nisi,
            pulvinar commodo nisl consequat eu. Suspendisse neque
            nisl, bibendum vitae lorem quis, tempor convallis
            mauris.
        </p>
    </div>
    <div id="vert">
        <img src="00-img.png" alt="description" width="180" height="
        285">
    </div>
</div>
<div id="bleu">bloc 3</div>
```

*Fig. 5*

Et le CSS, sur lequel on a supprimé les informations de hauteur pour **#vert** ainsi que la couleur de fond, ce qui nous donne au bout du compte :

```css
* {
    margin:0px;
    padding:0px;
}

body {
    width:600px;
    font: normal 12px arial;
    color:000;
    margin:25px auto;
}

#conteneur{
    background-color:#a6a6a6;
    border:5px solid #333;
    position:relative;
}

#jaune{
    background-color:#ccdd55;
    width:400px;
}

#vert {
    width:180px;
    position:absolute;
    top:0px;
    right:0;
}

#bleu {
    height:50px;
    background-color:#00aadd;
    border:5px solid #3ee;
}
```

*Fig. 6*

Le rendu :


*Fig. 7*

Maintenant, observons ce qu'il se produit s'il y a par exemple beaucoup moins de texte dans la partie gauche. Ici, on va uniquement modifier le HTML en retirant le premier paragraphe de texte. Le bloc **#jaune** devient alors :

```html
<div id="jaune" >
    <p>
        Lorem ipsum dolor sit amet, consectetur
        adipiscing elit. Maecenas euismod
        viverra neque, et ornare sapien rhoncus
        eu. Interdum et malesuada fames ac ante
        ipsum primis in faucibus. Maecenas vel
        sem luctus, dapibus nulla in, tincidunt
        diam. Mauris eleifend dui vel congue
        tristique. Morbi auctor, tortor et
        malesuada hendrerit, purus libero
        pretium lacus, et malesuada ligula erat
        eu ante. Cras cursus eget libero id
        fermentum. Ut vel lobortis quam. Donec
        venenatis porttitor sem, quis egestas
        diam laoreet rutrum. Donec feugiat,
        sapien ut consectetur pellentesque,
        magna mi dignissim ipsum, id varius
        turpis nisi vel arcu. Proin malesuada
        lacus faucibus tellus auctor, ac
        commodo dolor bibendum.
    </p>
</div>
```

*Fig. 8*

Le rendu :


*Fig. 9*

Comme l'élément vert est sorti du flux, il n'est plus pris en compte dans le calcul de la hauteur de la boîte **#conteneur**. On pourrait bien sûr donner une hauteur minimale à notre boîte de gauche afin de compenser l'éventuelle rareté de l'élément textuel, mais cela ne changera pas fondamentalement le problème, puisqu'on pourrait avoir une image encore plus grande en hauteur sur le côté droit.

Disons, pour résumer, que le danger du positionnement absolu intervient dans le cas où l'on n'est pas certain des hauteurs de nos boîtes gauche et droite. Dit autrement, il faut utiliser le positionnement absolu **si et seulement si** on est certain que la hauteur du bloc de gauche sera **toujours supérieure** à la hauteur du bloc de droite. Il n'y a en effet rien qui puisse empêcher ces débordements d'apparaître en conservant ce type de positionnement.

Que nous ayons le moindre doute quant aux valeurs des blocs **#droite** et **#gauche** et il nous faudra recourir aux flottements ou à ses équivalents. Mais comme nous allons le voir, nous ne sommes pas à l'abri de problèmes en recourant aux flottements. Il nous faut donc apprendre à les repérer pour mieux nous en prémunir.

## II. Flottements et débordements

Les cas de débordement en ce qui concerne les flottements sont au nombre de trois et ils ont tous la même cause, ce qui veut également dire qu'on peut leur appliquer une unique et même solution. Passons-les en revue.

### A. Cas 1

Le bloc vert flotte, il est hors du flux. Il se trouve dans le bloc jaune, les deux blocs étant eux-mêmes dans un bloc conteneur qui n'a pas beaucoup d'importance ici.

HTML

```html
<div id="conteneur">
    bloc conteneur
    <div class="jaune">bloc 1
        <div class="vert">bloc 2 qui déborde</div>
    </div>
</div>
```

*Fig. 10*

CSS

```css
* {
    margin:0px;
    padding:0px;
    box-sizing: border-box;
}

#conteneur{
    background-color:#ccc;
    margin:25px auto;
    padding:5px;
    width:600px;
    border:1px solid #ccc;
    min-height:200px;
    font: 24px arial;
    color:000;
    font-weight:bold;
}

#jaune{
    background-color:#ccdd55;
    min-height:330px;
    width:400px;
    float:left;
    margin-right:50px;
}

#vert {
    background-color:#55aa55;
    float:left;
    min-height:330px;
    width:130px;
}
```

*Fig. 11*

Ce qui a pour rendu :


*Fig. 12*

## B. Cas 2

Dans ce colonnage, les deux colonnes dépassent de leur conteneur.

HTML

```html
<div id="conteneur">
    <div id="jaune">colonne 1</div>
    <div id="vert"> colonne 2</div>
</div>
```

*Fig. 13*

CSS

```css
* {
    margin:0px;
    padding:0px;
    box-sizing: border-box;
}

#conteneur{
    background-color:#ededed;
    margin:25px auto;
    padding:5px;
    width:600px;
    border:1px solid #ccc;
    min-height:250px;
    font: 24px arial;
    color:000;
    font-weight:bold;
}

#jaune{
    background-color:#ccdd55;
    height:30px;
}

#vert {
    background-color:#55aa55;
    float:right;
    width:130px;
}
```

*Fig. 14*

Rendu :


*Fig. 15*

## C. Cas 3

Ici, un bloc qui suit des vignettes que l'on a fait flotter se retrouve perdu au milieu de celles-ci.

HTML

```html
<div id="conteneur">
    <div class="vert">V 1</div>
    <div class="vert">V 2</div>
    <div class="vert">V 3</div>
    <div class="vert">V 4</div>
    <div id="jaune">bloc jaune</div>
</div>
```

*Fig. 16*

CSS

```css
* {
    margin:0px;
    padding:0px;
    box-sizing: border-box;
}

#conteneur{
    background-color:#ccc;
    margin:25px auto;
    padding:5px;
    width:600px;
    border:1px solid #ccc;
    min-height:200px;
    font: 24px arial;
    color:000;
    font-weight:bold;
}

.vert {
    background-color:#55aa55;
    float:left;
    height:130px;
    width:130px;
    margin:25
}

#jaune {
    background-color:#ccdd55;
    min-height: 50px;
}
```

*Fig. 17*

Rendu :

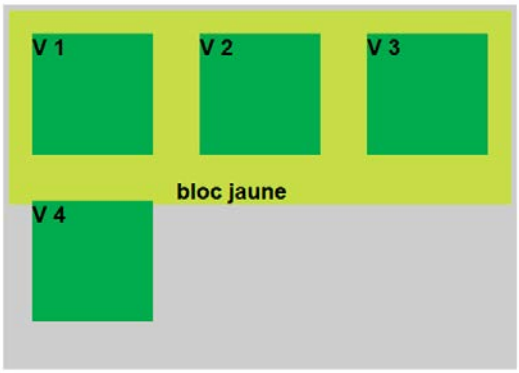


*Fig. 18*

## D. Diagnostic et solution

Normalement, si vous avez bien compris comment fonctionne le flottement, vous ne devriez pas avoir de problème pour résoudre ces dysfonctionnements.

Lorsqu'on regarde ce qu'il se passe chaque fois, c'est comme si des éléments n'étaient pas pris en compte dans le calcul des hauteurs. Ainsi dans le cas 1, la hauteur du bloc vert semble n'avoir aucun impact sur celle du bloc jaune. En 2, ce sont les deux colonnes jaunes et vertes qui semblent ne pas être prises en compte dans le calcul et le rendu de la hauteur de leur conteneur. Enfin, en 3, c'est comme si le bloc jaune ignorait que des vignettes vertes se trouvaient là.

Cela ne devrait pas nous surprendre puisque le propre des flottements est de sortir les éléments du flux normal. Plus tôt, on avait vu que l'une des grandes différences de comportement entre le positionnement relatif et absolu était que s'agissant du premier, on conservait la hauteur de l'élément en position relative dans la prise en compte de la hauteur du parent, même si cet élément ne se trouvait pas là.

Au contraire, dans la position absolue, l'élément est complètement sorti du flux et ne compte donc plus dans la hauteur des éléments qui se succèdent dans le flux normal. Le cas des flottements est à peu près le même à une différence près. Comme dit plus haut, le positionnement absolu n'a aucun moyen de résoudre par lui-même les débordements qu'il peut engendrer. On doit donc soit veiller à la taille des éléments en jeu, soit faire en sorte que cette taille soit toujours maîtrisée. Si l'on a le moindre doute, nous passerons par des flottements. Pour quelles raisons ?

C'est bien simple, dans le flottement également, nous sortons les éléments du flux. Mais cette sortie du flux normal n'est pas irrémédiable. Il suffit de remettre les éléments dans le flux **après le dernier élément flottant** pour que le flux reprenne son cours normal.

Au niveau du HTML, on va simplement rajouter notre célèbre et indispensable **.clear** (ou travailler sur le parent avec un **clearfix** et un ciblage avec la pseudo-classe **:after** ou encore l'**overflow:hidden** sur le parent).

Ce qui nous donne **pour le cas 1** :

HTML

```
<div id="conteneur">
    bloc conteneur
    <div class="jaune">bloc 1
        <div class="vert"> bloc 2 qui déborde </div>
        <div class="clear"></div>
    </div>
</div>
```

*Fig. 19*

CSS

On se contente d'ajouter aux règles CSS vues plus tôt l'unique règle dont on a besoin.

```
.clear {
    clear:both;
}
```

*Fig. 20*

Résultat :


*Fig. 21*

**Cas 2** :

HTML

```
<div id="conteneur">
    <div id="jaune">colonne 1</div>
    <div id="vert"> colonne 2</div>
    <div class="clear"></div>
</div>
```

*Fig. 22*

CSS

```
.clear {
    clear:both;
}
```

*Fig. 23*

Idem, on se contente d'ajouter le **clear**.

Résultat :


*Fig. 24*

**Cas 3** :

HTML

```
<div id="conteneur">
    <div class="vert">V 1</div>
    <div class="vert">V 2</div>
    <div class="vert">V 3</div>
    <div class="vert">V 4</div>
    <div class="clear" ></div>
    <!-- attention c'est bien après le dernier
    élément flottant que l'on doit placer le
    clear -->
    <div id="jaune">bloc jaune</div>
</div>
```

*Fig. 25*

CSS

```
.clear {
    clear:both;
}
```

*Fig. 26*

Idem, on se contente d'ajouter le **clear**.

Résultat :

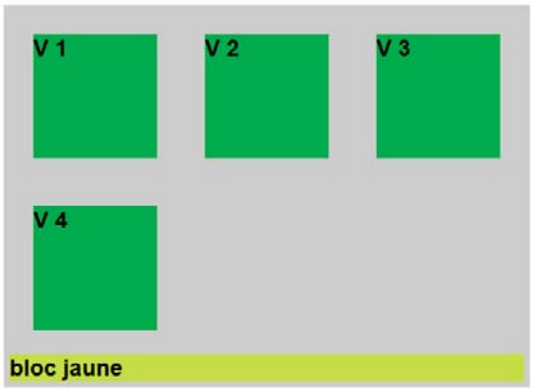


*Fig. 27*

## III. Conclusion

Au cours des derniers chapitres, nous avons abordé tour à tour différentes manières de sortir les éléments du flux par défaut, mais aussi étudié les complications que peuvent engendrer les uns ou les autres. Normalement, vous avez toutes les clés en main pour faire le bon choix lors de vos mises en page et éviter les problèmes que l'on rencontre le plus souvent. Néanmoins, il arrive que l'on oublie un détail et que l'on mette de longues minutes à identifier le problème. Avec le temps et l'expérience, ces désagréments se feront de plus en plus rares.