8 : LES ÉLÉMENTS FLOTTANTS

La propriété CSS **float** permet de sortir un élément du flux normal de la page et de le faire “flotter” contre un bord de son élément parent conteneur ou contre un autre élément flottant.

Une utilisation bien connue de **la propriété float** est de s’en servir pour faire flotter une image à droite ou à gauche d’un texte et ainsi l’entourer avec du texte.

La propriété float est donc une autre propriété qui va impacter la disposition dans la page et qu’il convient de manier avec précaution pour ne pas obtenir de comportement indésirable. Le but de cette nouvelle leçon est d’apprendre à la manipuler.

I. Définition et fonctionnement de la propriété float

La propriété **float** va retirer un élément du flux normal de la page puis le placer contre un bord de son élément parent (ou élément conteneur) ou contre le bord d’un élément flottant le précédent.

Cette propriété va également impacter les éléments environnants puisque le texte et les éléments **inline** suivants un élément possédant un **float** différent de **none** vont essayer de venir se placer à ses côtés.

La propriété **float** était à l’origine principalement utilisée pour incruster des images dans du texte en les faisant flotter. L’utilisation « normale » de **float** est donc d’appliquer le **float** (de faire flotter) des éléments **inline** dans la page et de laisser les textes se positionner autour.

En effet, les éléments de type **block** suivants un élément flottant vont également se placer sur la même ligne que l’élément flottant mais continuer à prendre tout l’espace disponible dans la ligne. La partie des éléments **block** chevauchant un flottant va être cachée derrière le flottant.

On va cependant également tout à fait pouvoir faire flotter un élément **block** même si généralement nous utiliserons plutôt un **display : inline-block** pour placer deux éléments de type **block** côte-à-côte.

Notez que si vous voulez faire flotter un élément **block** sans contenu, alors il faudra lui donner une largeur et une hauteur explicites avec les propriétés **width** et **height** car dans le cas contraire les valeurs calculées seront égales à 0.

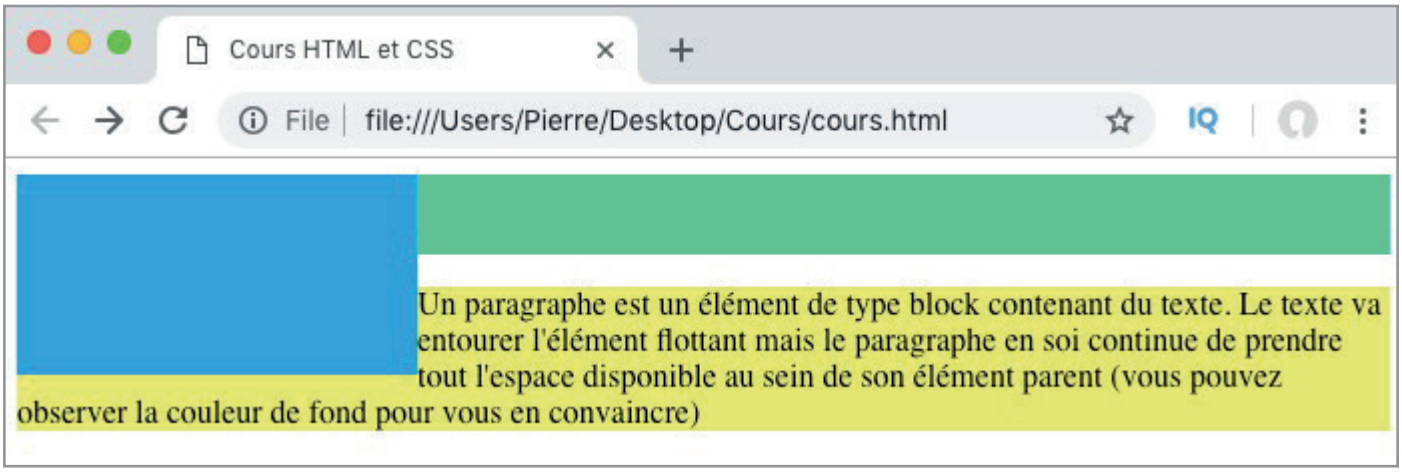
```html
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset= "utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <div class="flottant"></div>
        <div></div>
        <p>Un paragraphe est un élément de type block contenant du texte. Le
        texte va entourer l'élément flottant mais le paragraphe en soi continue
        de prendre tout l'espace disponible au sein de son élément parent
        (vous pouvez observer la couleur de fond ou inspecter l'élément
        pour vous en convaincre)</p>
    </body>
</html>
```

index.html

```css
.flottant{
  width: 200px;
  height: 100px;
  float: left;
  background-color: #0AF;/*Bleu*/
}
div{
  height: 40px;
  background-color: #0FA;/*Vert*/
}
p{
  background-color: #EE6;/*Jaune*/
}
```

cours.css

Résultat :



Notez également déjà que la propriété **float** ne va pas fonctionner avec des éléments positionnés de manière absolue et ne va avoir aucun effet sur des éléments affichés avec **display : flex** ou **display : inline-flex**. Nous aurons l'occasion de revenir sur ces sujets plus tard.

II. Les valeurs de la propriété float

Historiquement, nous avions le choix entre 3 valeurs à passer à la propriété **float** :

- **float : left :** L'élément va venir se positionner à l'extrémité gauche de son élément conteneur ou va être décalé sur la gauche jusqu'à toucher un autre élément avec **float : left ;**

- **float : right :** L'élément va venir se positionner à l'extrémité droite de son élément conteneur ou va être décalé sur la droite jusqu'à toucher un autre élément avec **float : right ;**

- **float : none :** Valeur par défaut. L'élément n'est pas un élément flottant.

Le CSS3 va apporter un nouveau choix élargi de valeurs que nous allons pouvoir passer à **float**.
Je vous rappelle ici que le CSS3 est toujours en développement et qu'ainsi tout ce qui est en train d'être établi par celui-ci n'est pas forcément encore passé comme recommandation du W3C. En effet, certaines valeurs et propriétés actuellement à l'étude dans le cadre du CSS3 vont potentiellement être modifiées ou abandonnées en cours de route.

Parmi les nouvelles valeurs apportées à **float**, cependant, nous pouvons déjà en citer deux qui possèdent déjà un bon support par les navigateurs :

- **float : inline-start :** L'élément va venir se positionner au début de son élément conteneur (c'est-à-dire à gauche pour des documents dont l'écriture se fait de gauche à droite ou à droite dans le cas contraire) ou va être décalé vers le début de son conteneur jusqu'à toucher un autre élément avec **float : inline-start ;**

- **float : inline-end :** L'élément va venir se positionner à la fin de son élément conteneur (c'est-à-dire à droite pour des documents dont l'écriture se fait de gauche à droite ou à gauche dans le cas contraire) ou va être décalé vers la fin de son conteneur jusqu'à toucher un autre élément avec **float : inline-end ;**

Float : none
La valeur par défaut de float est none. Cette valeur correspond à l'absence de flottement.

Exemple pour manipuler les valeurs :

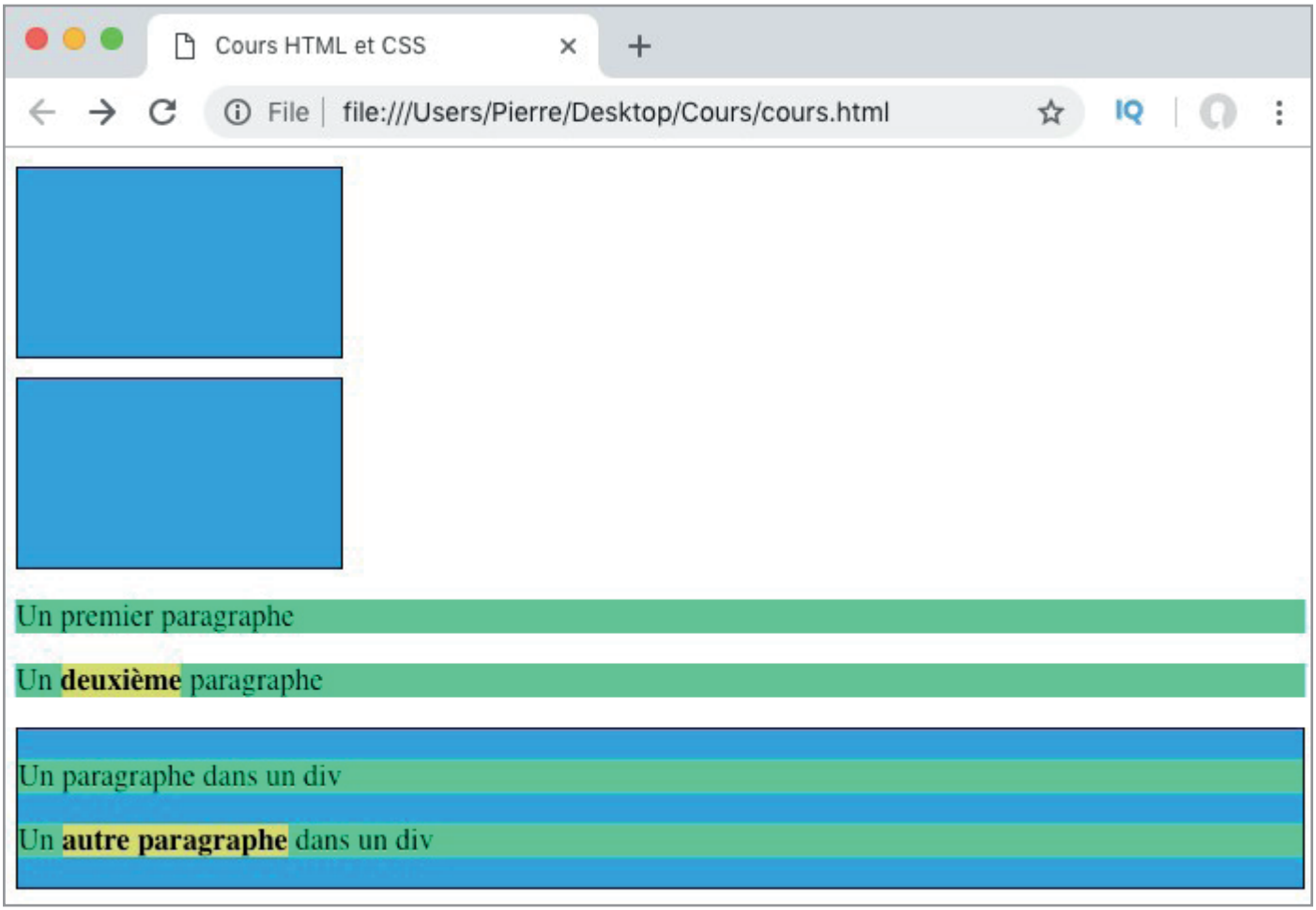
```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset= "utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <div class="w25 h100"></div>
        <div class="w25 h100"></div>
        <p>Un premier paragraphe</p>
        <p>Un <strong>deuxième</strong> paragraphe</p>
        <div>
            <p>Un paragraphe dans un div</p>
            <p>Un <strong>autre paragraphe</strong> dans un div</p>
        </div>
    </body>
</html>
```

index.html

```css
div{
  background-color: #0AF;/*Bleu*/
  border: 1px solid black;
  margin: 10px 0px;/*10px de marge haute et basse*/
}
p{
  background-color: #0FA;/*Vert*/
}
strong{
  background-color: #DD6;/*Jaune*/
}

.w25{
  width: 25%;
}
.h100{
  height: 100px;
}
```

cours.css

Résultat :



Ici, nous avons deux éléments **div** de largeurs égales à **25%** de celle de leur parent et de hauteurs égales à **50px**. Les **div** sont des éléments de type **block** et vont donc aller à la ligne et occuper une ligne chacun quelle que soit leurs dimensions.
Ensuite, nous avons deux éléments **p** dont un qui contient un élément **strong**. Les éléments p sont également de type **block** et vont donc par défaut occuper tout l'espace disponible dans leur parent et occuper une ligne chacun. L'élément **strong** est lui un élément **inline** par défaut et va donc ne prendre que la place nécessaire dans son parent et ne pas aller à la ligne.

Finalement, nous avons créé un **div** qui contient deux paragraphes dont un contient lui-même un autre élément **strong**.

Par défaut, aucun de ces éléments ne flotte. Ne rien préciser ou préciser un **float : none** est ici identique et le comportement de ces éléments est connu.

Note : parfois, il sera utile de préciser un **float : none** pour un élément pour annuler par exemple un comportement d'héritage.

Float : left et float : right

En appliquant un **float: left** à un élément, l'élément va venir se positionner contre le bord gauche de son élément conteneur ou va être décalé vers la gauche jusqu'à toucher un autre élément flottant.

En appliquant un **float: right** à un élément, l'élément va venir se positionner contre le bord droit de son élément conteneur ou va être décalé vers la droite jusqu'à toucher un autre élément flottant.

Ces deux valeurs vont se comporter et pouvoir être appliquées de manière similaire.

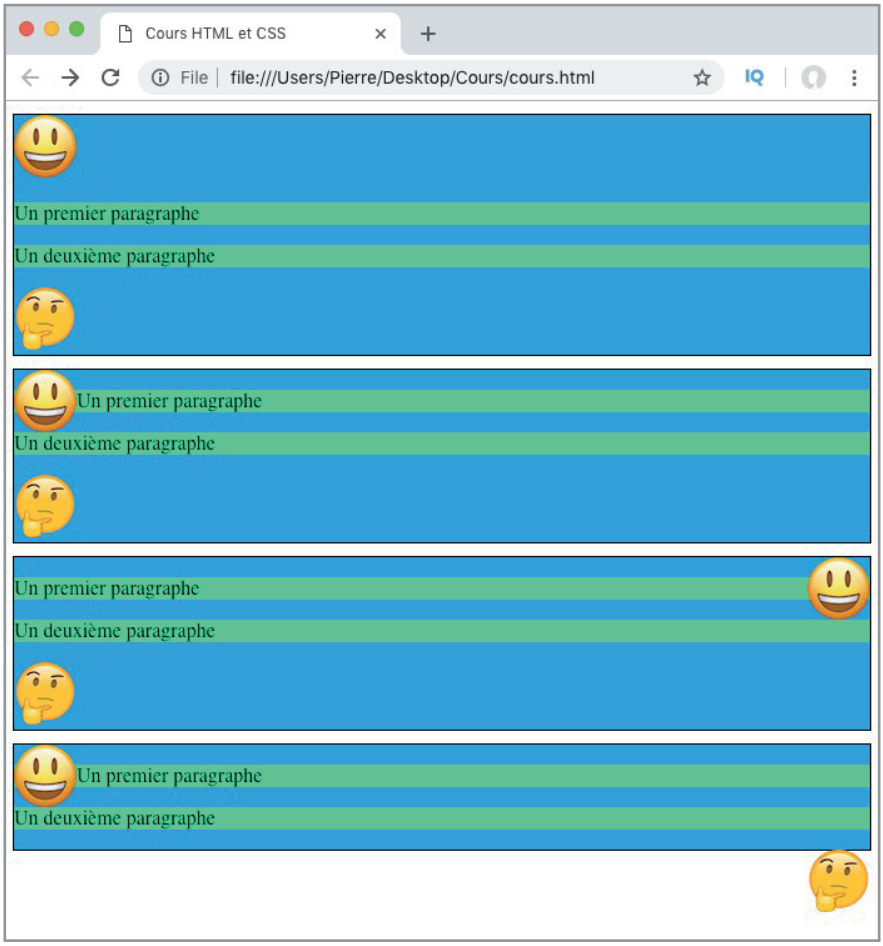
```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset= "utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <div class="conteneur">
            <img src="smile.png" alt="Un emoji content">
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
            <img src="think.png" alt="Un emoji penseur">
        </div>
        <div class="conteneur">
            <img src="smile.png" alt="Un emoji content" class="gauche">
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
            <img src="think.png" alt="Un emoji penseur">
        </div>
        <div class="conteneur">
            <img src="smile.png" alt="Un emoji content" class="droite">
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
            <img src="think.png" alt="Un emoji penseur">
        </div>
        <div class="conteneur">
            <img src="smile.png" alt="Un emoji content" class="gauche">
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
            <img src="think.png" alt="Un emoji penseur" class="droite">
        </div>
    </body>
</html>
```

index.html

```css
div{
  background-color: #0AF;/*Bleu*/
  border: 1px solid black;
  margin: 10px 0px;/*10px de marge haute et basse*/
}
p{
  background-color: #0FA;/*Vert*/
}
strong{
  background-color: #DD6;/*Jaune*/
}

.w25{
  width: 25%;
}
.h100{
  height: 100px;
}

.gauche{
  float: left;
}
.droite{
  float: right;
}
```

cours.css

Résultat :



Il y a des choses intéressantes à noter dans cet exemple. Ici, on va faire flotter nos différentes images d'emoji à droite ou à gauche. La première chose à retenir est qu'un élément flottant va toujours flotter sur sa ligne. Dans le dernier exemple, par exemple, notre premier emoji est le premier élément déclaré dans notre conteneur, il flottera donc en haut tandis que notre deuxième emoji est le dernier élément déclaré et il flottera donc en bas du conteneur.

Dans notre dernier **div**, en particulier, l'emoji « penseur » se retrouve en dehors du div. Cela est dû au fait que **float** retire l'élément du flux normal de la page : le div ne tiendra plus compte de cet élément et va donc se redimensionner à la taille des éléments qu'il contient, c'est-à-dire dans ce cas les deux paragraphes. L'élément flotté va lui en revanche continuer à flotter sur sa ligne.

Si vous regardez de plus près, vous pouvez constater qu'il se passe exactement la même chose avec notre premier emoji mais c'est moins visible car c'est le premier élément déclaré dans notre div et donc il va toujours être inclus dedans grâce aux paragraphes le suivant.

Cependant, ne vous y trompez pas : le **div** ne tient plus compte de l'emoji flotté dans le calcul de sa taille mais seulement des éléments non flottés et va se redimensionner en conséquence. C'est la raison pour laquelle nos deuxième et troisième **div** sont plus petits en hauteur que le premier. Pour éviter ce genre de rendu visuel, on peut toujours augmenter artificiellement la taille de notre élément conteneur en utilisant **height**.

La deuxième chose à bien comprendre est le fait qu'un élément flottant n'impacte pas les propriétés des éléments environnants à proprement parler : les éléments suivants un élément flottant vont pouvoir se placer à côté de l'élément flottant dans la limite de la hauteur de l'élément flottant. Les éléments sous l'élément flottant vont avoir un comportement « normal » comme on peut le voir avec notre deuxième paragraphe pour nos deuxième et troisième **div**.

III. Contrôler le comportement des éléments autour d'un flottant avec la propriété clear

La propriété CSS clear va nous permettre d'empêcher un élément de se positionner à côté d'un élément flottant. Cette propriété va être extrêmement utile dans de nombreuses situations pour mieux contrôler le design de nos pages.

Nous allons pouvoir lui passer l'une des valeurs suivantes :

- **clear : none :** Valeur par défaut. Laisse les éléments se positionner à côté d'éléments flottants ;
- **clear : left :** Empêche un élément de se positionner à côté d'éléments possédant un float : left ;
- **clear : right :** Empêche un élément de se positionner à côté d'éléments possédant un float : right ;
- **clear : both :** Empêche un élément de se positionner à côté d'éléments possédant un float : left ou un float : right ;
- **clear : inline-start :** Empêche un élément de se positionner à côté d'éléments possédant un float : inline-start ;
- **clear : inline-end :** Empêche un élément de se positionner à côté d'éléments possédant un float : inline-end.

Notez ici que la propriété **clear** va avoir un comportement légèrement différent selon qu'on l'applique à des éléments non flottants ou au contraire à des éléments déjà flottants.

Dans le cas où l'on applique **clear** à un élément non flottant, l'élément va être déplacé de telle sorte à ce que sa bordure supérieure se place directement sous le bord de la marge basse extérieure des éléments flottants concernés et il va y avoir fusion des marges (collapse en anglais).

Dans le cas où **clear** est appliquée à un élément flottant, l'élément va être déplacé de telle sorte à ce que l'extrémité de sa marge supérieure se place directement sous le bord de la marge basse des éléments flottants concernés. Les deux marges vont donc être conservées. La position des potentiels éléments flottants suivants va être impactée puisqu'un élément flottant ne peut pas être situé plus haut qu'un autre élément flottant qui le précède.

Cas pratiques d'utilisation de la propriété clear :

Généralement, nous allons donc **utiliser la propriété clear après avoir appliqué un float** à un élément pour empêcher certains éléments de venir flotter à côté de l'élément en question.

En effet, bien souvent, lorsque nous créerons le design d'une page, nous n'allons pas vouloir que les éléments se positionnent tant qu'ils le peuvent à côté d'un élément flottant mais pouvoir contrôler quels éléments vont pouvoir se positionner aux côtés d'un élément flottant et quels éléments ne doivent pas le faire.

Illustration du problème réglé par la propriété clear :

Pour bien comprendre l'intérêt de la propriété float, je vous propose de regarder l'exemple suivant :

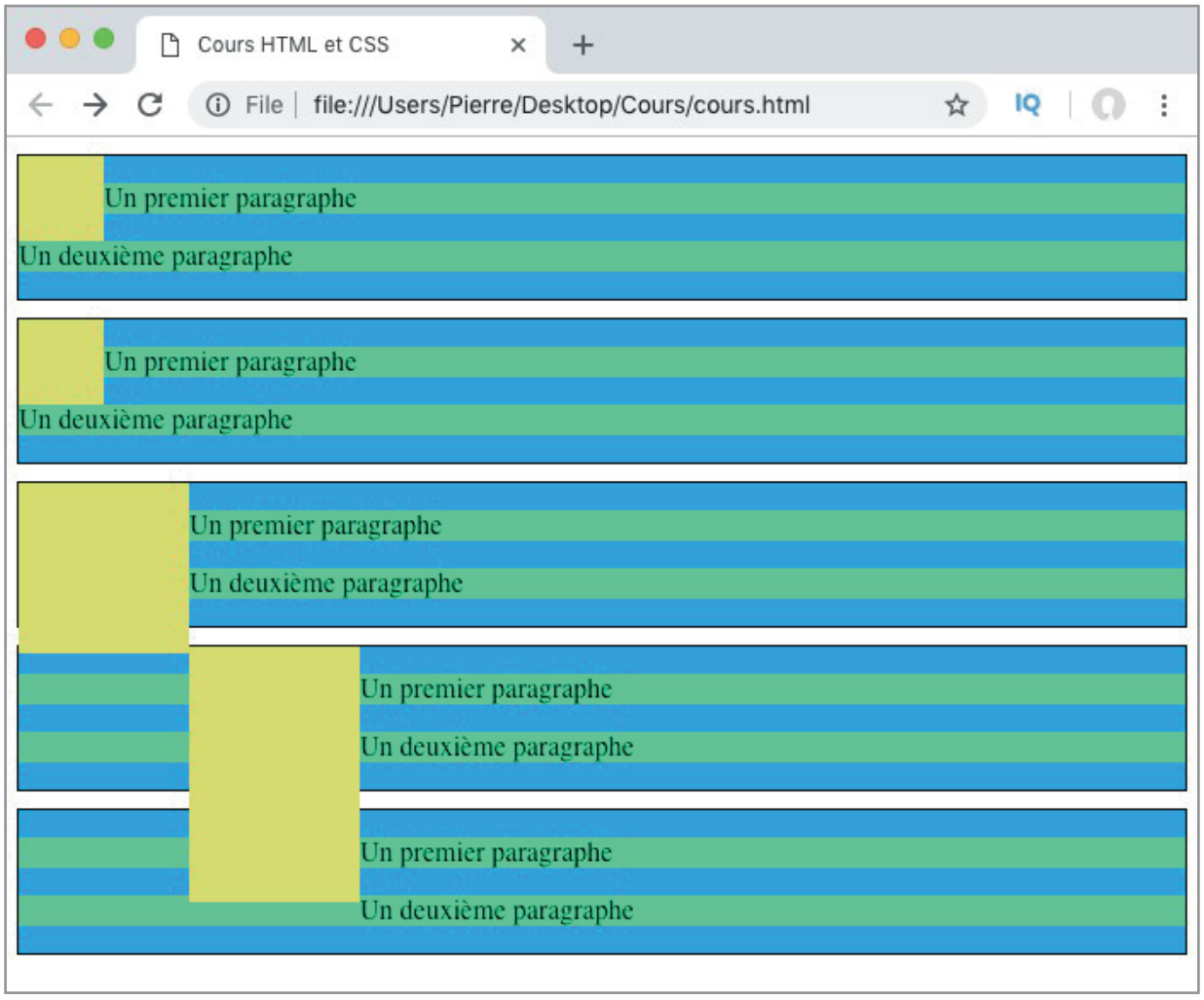
```html
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset= "utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <div class="conteneur">
            <div class="interne gauche w50 h50"></div>
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
        </div>
        <div class="conteneur">
            <div class="interne gauche w50 h50"></div>
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
        </div>
        <div class="conteneur">
            <div class="interne gauche w100 h100"></div>
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
        </div>
        <div class="conteneur">
            <div class="interne gauche w100 h150"></div>
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
        </div>
        <div class="conteneur">
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
        </div>
    </body>
</html>
```

index.html

```css
p{
  background-color: #0FA;/*Vert*/
}
.conteneur{
  background-color: #0AF;/*Bleu*/
  border: 1px solid black;
  margin: 10px 0px;/*10px de marge haute et basse*/
}
.interne{
  background-color: #DD6;/*Jaune*/
}
.w50{width: 50px;}
.w100{width: 100px;}
.h50{height:50px;}
.h100{height:100px;}
.h150{height:150px;}

.gauche{
  float: left;
}
.droite{
  float: right;
}
```

cours.css

Résultat :



Ici, nous faisons flotter chacun des **div internes** à gauche dans nos **div conteneurs**. Pour nos deux premiers div, le résultat est celui espéré. En revanche, on observe un décalage qui se crée pour nos trois derniers div.

En fait, c'est tout à fait normal : le fait que l'affichage se fasse bien avec nos deux premiers div est simplement un « coup de chance » dû au fait que nous n'avons pas deux éléments flottants qui se touchent puisque les div conteneurs soient plus grands que les div flottés et qu'ils ne contiennent qu'un élément flottant.

Ici, il faut savoir que par défaut tous les éléments suivants un flottant vont essayer de se positionner sur la même ligne que l'élément flottant et cela dans la limite de la hauteur de l'élément flottant. Dans nos deux premiers exemples, le deuxième paragraphe va à la ligne tout simplement car il n'a pas la place pour se positionner sur la même ligne que le flottant (la hauteur du flottant est déjà remplie par le premier paragraphe).

Pour les div flottants suivants, en revanche, la situation va être différente. Notre troisième div flottant dépasse en effet de son div conteneur et va arriver jusqu'à la ligne sur laquelle se situe le div flottant suivant.

Le quatrième div flottant va donc toucher le div flottant précédent. Or, un élément flottant va essayer se positionner soit contre le bord de son élément parent soit contre le bord d'un élément flottant précédent si un tel élément existe.

C'est ce qui se passe ici : notre dernier div flotté va rencontrer le div flotté précédent et va donc se coller contre son bord.

Enfin, notre quatrième et dernier div flottant est suffisamment haut pour arriver jusqu'aux lignes occupées par les paragraphes du dernier div conteneur. Le texte des paragraphes va donc se positionner contre le flottant du div conteneur précédent.

La propriété clear va justement nous permettre d'éviter ce genre de comportements généralement non souhaités.

Ici, par exemple, il suffirait d'appliquer un **clear : left** à chacun de nos div conteneurs pour résoudre une grande partie du problème :

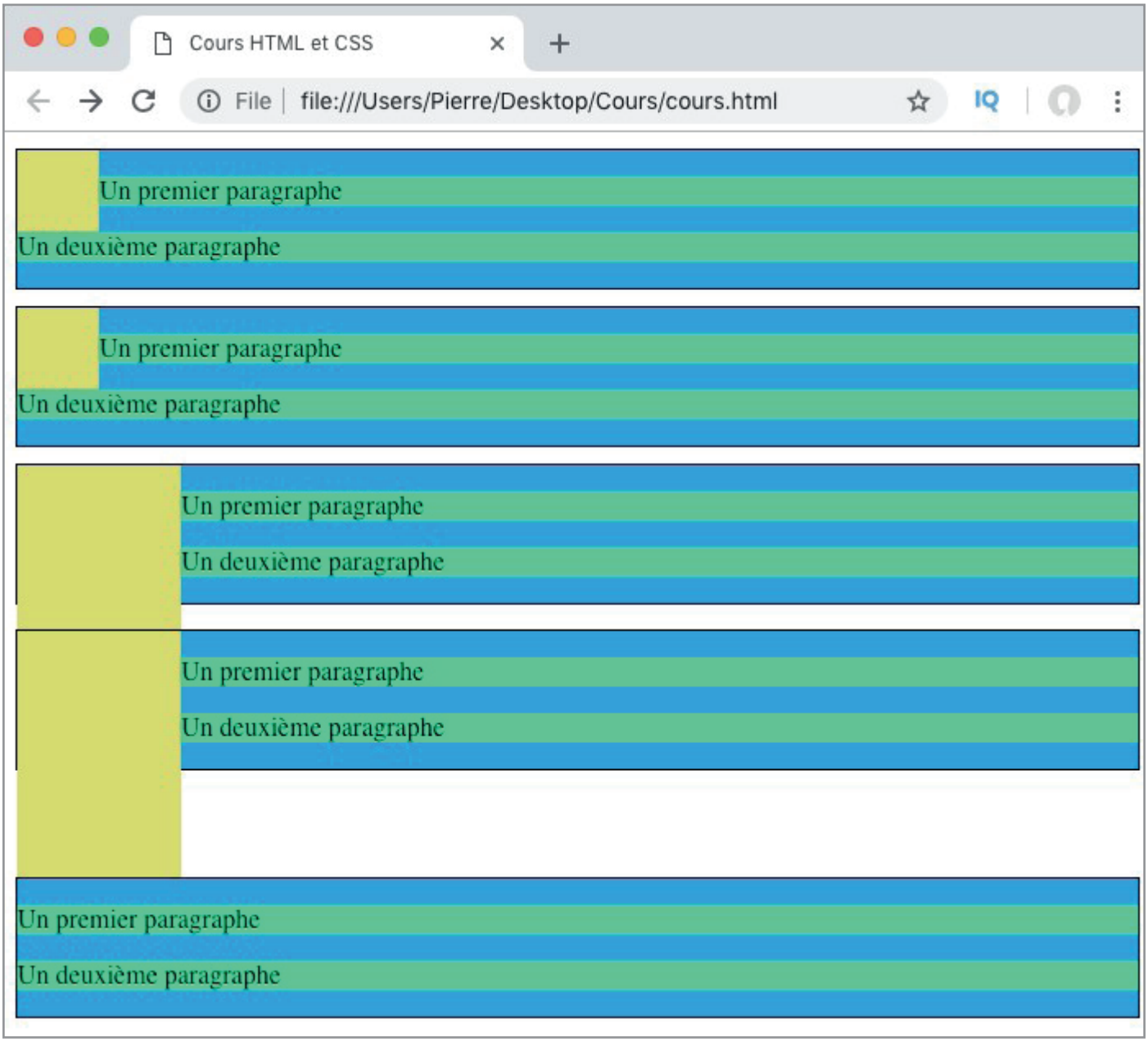
```css
p{
  background-color: #0FA;/*Vert*/
}
.conteneur{
  background-color: #0AF;/*Bleu*/
  border: 1px solid black;
  margin: 10px 0px;/*10px de marge haute et basse*/
}
.interne{
  background-color: #DD6;/*Jaune*/
}
.w50{width: 50px;}
.w100{width: 100px;}
.h50{height:50px;}
.h100{height:100px;}
.h150{height:150px;}

.gauche{
  float: left;
}
.droite{
  float: right;
}

.cleared{
  clear: left;
}
```

cours.css

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset= "utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <div class="conteneur">
            <div class="interne gauche w50 h50"></div>
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
        </div>
        <div class="conteneur cleared">
            <div class="interne gauche w50 h50"></div>
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
        </div>
        <div class="conteneur cleared">
            <div class="interne gauche w100 h100"></div>
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
        </div>
        <div class="conteneur cleared">
            <div class="interne gauche w100 h150"></div>
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
        </div>
        <div class="conteneur cleared">
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
        </div>
    </body>
</html>
```

index.html

Résultat :



IV. Découverte et utilisation du « clearfix » CSS

Il nous reste un problème à régler : comment faire lorsque le flottant est le dernier élément dans son conteneur pour ne pas que celui-ci dépasse du conteneur ?

En effet, bien souvent, on voudra que les éléments flottants restent dans la limite de leur conteneur ou plus exactement que les conteneurs s'adaptent pour inclure les flottants dans leur taille.

Pour faire cela, on va utiliser ce qu'on appelle un **« clearfix »** qui est un hack bien connu en CSS utilisant le pseudo-élément **::after**.

Attention : il faudra cette fois-ci l'appliquer au conteneur qui contient le flottant qui pose des problèmes de design pour qu'il fonctionne. Ce clearfix est le suivant :

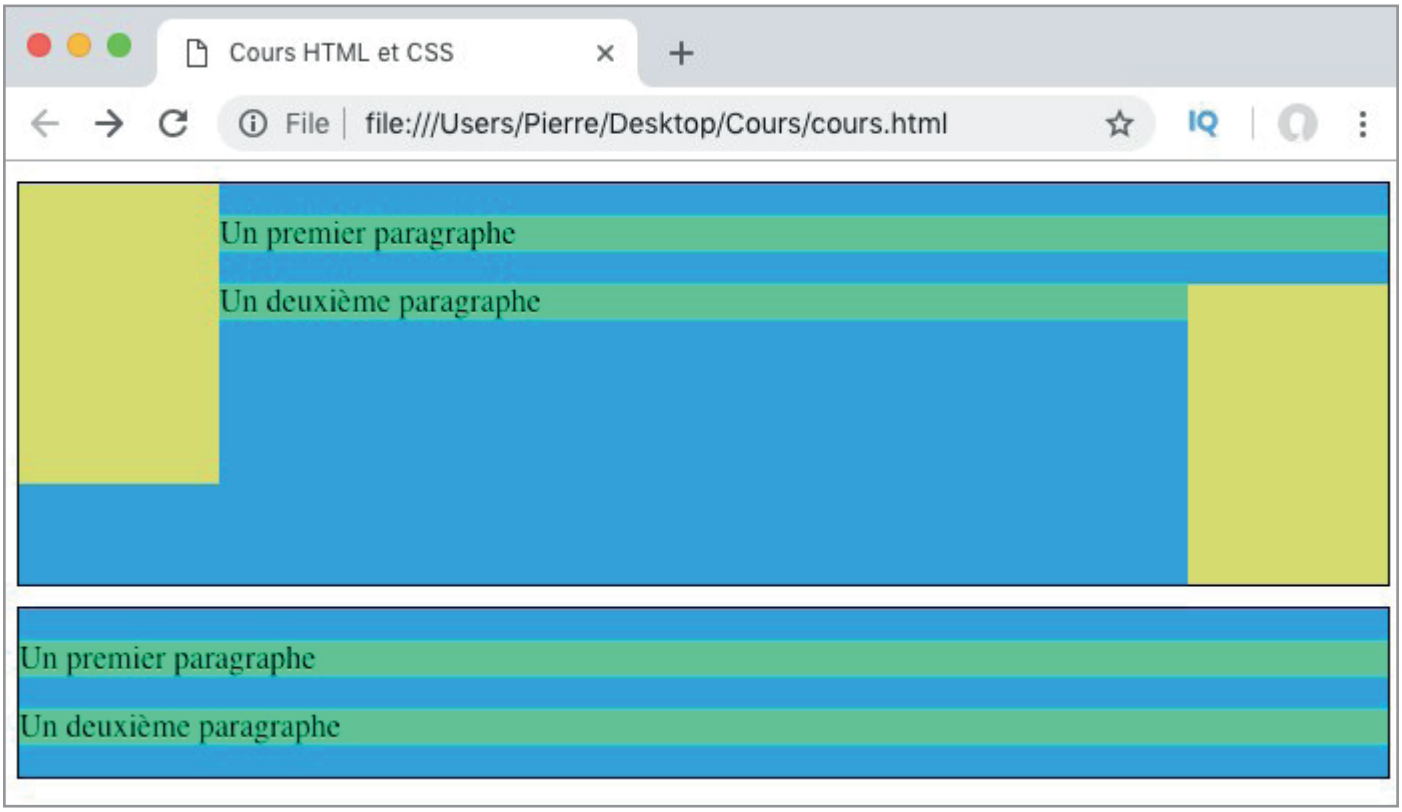
```html
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset= "utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <div class="conteneur clearfix">
            <div class="interne gauche w100 h150"></div>
            <p>Un premier paragraphe</p>
            <div class="interne droite w100 h150"></div>
            <p>Un deuxième paragraphe</p>
        </div>
        <div class="conteneur">
            <p>Un premier paragraphe</p>
            <p>Un deuxième paragraphe</p>
        </div>
    </body>
</html>
```

index.html

```css
p{
  background-color: #0FA;/*Vert*/
}
.conteneur{
  background-color: #0AF;/*Bleu*/
  border: 1px solid black;
  margin: 10px 0px;/*10px de marge haute et basse*/
}
.interne{
  background-color: #DD6;/*Jaune*/
}
.w100{width: 100px;}
.h150{height:150px;}

.gauche{
  float: left;
}
.droite{
  float: right;
}

.clearfix::after{
  content: "";
  display: table;
  clear: both;
}
```

cours.css

Résultat :



Nous étudierons les pseudo-éléments plus tard dans ce cours. Cependant, je vais quand même essayer de vous expliquer brièvement comment fonctionne le clearfix. Le pseudo élément **::after** nous permet de créer un pseudo élément qui va être le dernier enfant de l'élément sélectionné.

L'idée est ici de l'utiliser avec la propriété **content** pour ajouter un contenu ou plus exactement une chaine de caractères vide en fin de l'élément.

Ensuite, nous allons appliquer le **clear** à ce pseudo élément. Pour que cela fonctionne, cependant, il va falloir lui définir un **display**. Le type d'affichage qui se prête le mieux à l'opération est ici **display : table ;**

V. La propriété float et la hauteur des conteneurs

Comme nous avons pu le voir et l'évoquer plus haut, un élément flottant est retiré du flux normal de la page.

Cela implique que l'élément qui contient le flottant ne va pas tenir compte de ce dernier dans le calcul de ses dimensions.

Cela peut donc mener à des problèmes de dépassement du flottant de son élément conteneur comme on a pu le voir précédemment.

Il y a un cas que nous n'avons cependant pas encore étudié : le cas où le conteneur ne contient que des éléments flottants. Dans ce cas-là, la hauteur du conteneur va être égale à la taille de ses bordures et de son padding et donc nulle si le conteneur ne possède ni bordures ni marges internes.

Ce comportement sera rarement souhaité. Pour rétablir la hauteur du conteneur, il suffit une nouvelle fois d'utiliser le clearfix vu dans le chapitre précédent.

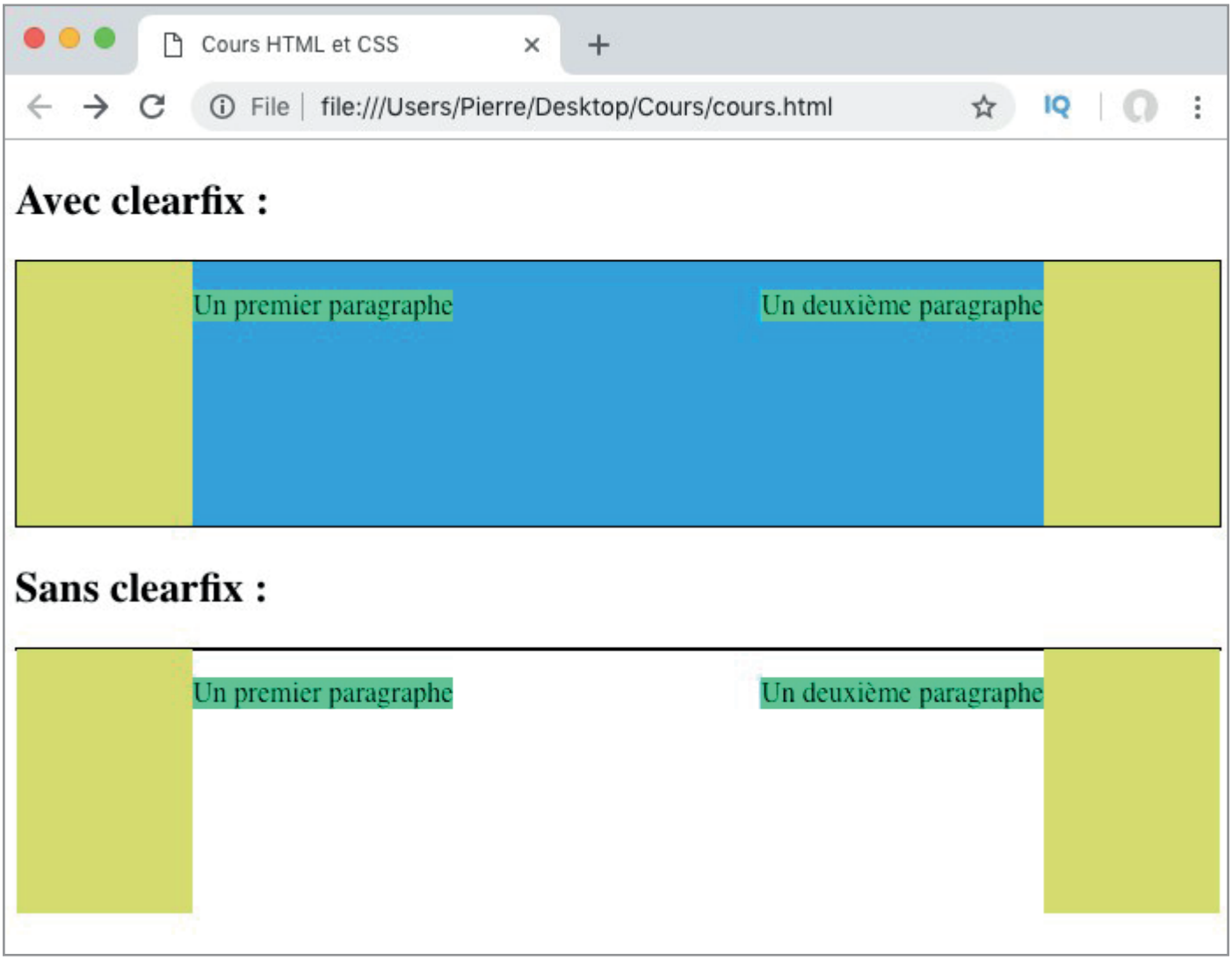
```html
<!DOCTYPE html>
<html>
    <head>
        <title>Cours HTML et CSS</title>
        <meta charset= "utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h2>Avec clearfix : </h2>
        <div class="conteneur clearfix">
            <div class="interne gauche w100 h150"></div>
            <p class="gauche">Un premier paragraphe</p>
            <div class="interne droite w100 h150"></div>
            <p class="droite">Un deuxième paragraphe</p>
        </div>

        <h2>Sans clearfix :</h2>
        <div class="conteneur">
            <div class="interne gauche w100 h150"></div>
            <p class="gauche">Un premier paragraphe</p>
            <div class="interne droite w100 h150"></div>
            <p class="droite">Un deuxième paragraphe</p>
        </div>
    </body>
</html>
```

index.html

```css
p{
  background-color: #0FA;/*Vert*/
}
.conteneur{
  background-color: #0AF;/*Bleu*/
  border: 1px solid black;
  margin: 10px 0px;/*10px de marge haute et basse*/
}
.interne{
  background-color: #DD6;/*Jaune*/
}
.w100{width: 100px;}
.h150{height:150px;}

.gauche{
  float: left;
}
.droite{
  float: right;
}

.clearfix::after{
  content: "";
  display: table;
  clear: both;
}
```

cours.css

Résultat :


Cours HTML et CSS
File | file:///Users/Pierre/Desktop/Cours/cours.html
Avec clearfix :
Un premier paragraphe
Un deuxième paragraphe
Sans clearfix :
Un premier paragraphe
Un deuxième paragraphe