Apprenez à créer votre site web avec HTML5 et CSS3

20 heures Facile Licence

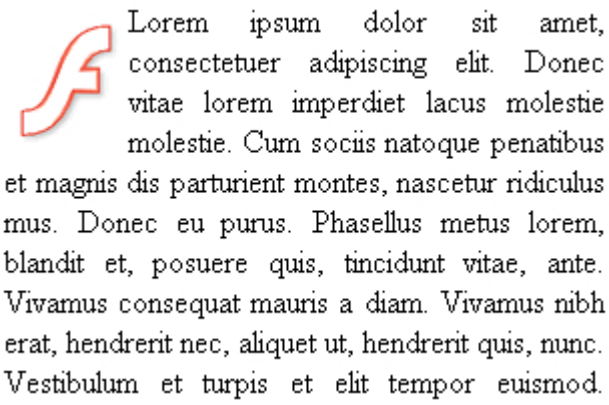
QUELQUES AUTRES TECHNIQUES DE MISE EN PAGE

Aujourd'hui, Flexbox est l'outil de mise en page que je recommande... mais vous devriez toujours connaître les autres techniques de mise en page que je vais vous présenter dans ce chapitre. Plus anciennes, elles ont l'avantage d'être reconnues par les vieilles versions des navigateurs. Enfin, si vous êtes amenés à gérer un "vieux" code, vous aurez besoin de connaître ces techniques de positionnement !

Ce chapitre vous sera donc plus probablement utile si vous avez besoin de gérer de vieux sites. Si vous êtes sur un nouveau projet, je recommande d'utiliser à la place une mise en page à base de Flexbox !

Le positionnement flottant

Vous vous souvenez de la propriété `float` ? Nous l'avons utilisée pour faire flotter une image autour du texte (figure suivante).



L'image flotte autour du texte grâce à la propriété float

Il se trouve que cette propriété est aujourd'hui utilisée par la majorité des sites web pour… faire la mise en page ! En effet, si on veut placer son menu à gauche et le contenu de sa page à droite, c'est *a priori* un bon moyen. Je dis bien *a priori* car, à la base, cette propriété n'a pas été conçue pour faire la mise en page et nous allons voir qu'elle a quelques petits défauts.

Reprenons le code HTML structuré que nous avons réalisé il y a quelques chapitres :

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <title>Zozor - Le Site Web</title>
    </head>

    <body>
        <header>
            <h1>Zozor</h1>
            <h2>Carnets de voyage</h2>
        </header>

        <nav>
            <ul>
                <li><a href="#">Accueil</a></li>
                <li><a href="#">Blog</a></li>
                <li><a href="#">CV</a></li>
            </ul>
        </nav>

        <section>
            <aside>
```

```
25                  <h1>À propos de l'auteur</h1>
26                  <p>C'est moi, Zozor ! Je suis né un 23 novembre 2005.</p>
27              </aside>
28              <article>
29                  <h1>Je suis un grand voyageur</h1>
30                  <p>Bla bla bla bla (texte de l'article)</p>
31              </article>
32          </section>
33
34          <footer>
35              <p>Copyright Zozor - Tous droits réservés
36
37              <a href="#">Me contacter !</a></p>
38          </footer>
39
40      </body>
41  </html>
```

Je rappelle que, sans CSS, la mise en page ressemble à la figure suivante.

Une page HTML sans CSS

Nous allons essayer de placer le menu à gauche et le reste du texte à droite. Pour cela, nous allons faire flotter le menu à gauche et laisser le reste du texte se placer à sa droite.

Nous voulons que le menu occupe 150 pixels de large. Nous allons aussi rajouter une bordure noire autour du menu

et une bordure bleue autour du corps (à la balise `<section>`) pour bien les distinguer :

```css
nav
{
    float: left;
    width: 150px;
    border: 1px solid black;
}

section
{
    border: 1px solid blue;
}
```

Voici le résultat à la figure suivante. Ce n'est pas encore tout à fait cela.

Le menu est bien positionné mais collé au texte

Il y a deux défauts (mis à part le fait que c'est encore bien moche) :

- Le texte du corps de la page touche la bordure du menu. Il manque une petite marge…
- Plus embêtant encore : la suite du texte passe… sous le menu !

On veut bien que le pied de page (« Copyright Zozor ») soit placé en bas sous le menu mais, par contre, on aimerait que tout le corps de page soit constitué d'un seul bloc placé à droite.

Pour résoudre ces deux problèmes d'un seul coup, il faut ajouter une marge extérieure à gauche de notre `<section>`, marge qui doit être par ailleurs *supérieure* à la largeur du menu. Si notre menu fait 150 px, nous allons par exemple donner une marge extérieure gauche de 170 px à notre section de page (figure suivante), ici à la ligne 10.

css

```
1  nav
2  {
3      float: left;
4      width: 150px;
5      border: 1px solid black;
6  }
7
8  section
9  {
10     margin-left: 170px;
11     border: 1px solid blue;
12 }
```

Le corps de page est bien aligné à droite du menu

Voilà, le contenu de la page est maintenant correctement aligné.

À l'inverse, vous pouvez aussi préférer qu'un élément se place obligatoirement sous le menu. Dans ce cas, il faudra utiliser… `clear: both;`, que nous avions déjà découvert, qui oblige la suite du texte à se

positionner sous l'élément flottant.

Transformez vos éléments avec display

Je vais vous apprendre ici à modifier les lois du CSS (brrr…). Accrochez-vous !

Il existe en CSS une propriété très puissante : `display`. Elle est capable de *transformer* n'importe quel élément de votre page d'un type vers un autre. Avec cette propriété magique, je peux par exemple imposer à mes liens (originellement de type inline) d'apparaître sous forme de blocs :

```css
a
{
    display: block;
}
```

À ce moment-là, les liens vont se positionner les uns en-dessous des autres (comme des blocs normaux) et il devient possible de modifier leurs dimensions !

Voici quelques-unes des principales valeurs que peut prendre la propriété `display` en CSS (il y en a encore d'autres) :

Valeur	Exemples	Description
`inline`	`<a>`, `<em>`, `<span>`…	Eléments d'une ligne. Se placent les uns à côté des autres.
`block`	`<p>`, `<div>`, `<section>`…	Eléments en forme de blocs. Se placent les uns en-dessous des autres et peuvent être redimensionnés.
`inline-block`	`<select>`, `<input>`	Eléments positionnés les uns à côté des autres (comme les inlines) mais qui peuvent être redimensionnés (comme les blocs).
`none`	`<head>`	Eléments non affichés.

On peut donc décider de masquer complètement un élément de la page avec cette propriété. Par exemple, si je veux masquer les éléments qui ont la classe « secret », je vais écrire :

```css
.secret
{
    display: none;
}
```

Pour faire apparaître ces éléments par la suite, vous devrez faire appel à JavaScript. Certains sites web utilisent cette technique pour, par défaut, masquer les sous-menus qui ne s'affichent que lorsqu'on parcourt les menus.

Et quel est ce nouveau type bizarre, `inline-block` ? C'est un mélange ?

Oui, ce type d'élément est en fait une combinaison des inlines et des blocs. C'est un peu le meilleur des deux mondes : les éléments s'affichent côte à côte et peuvent être redimensionnés.

Peu de balises sont affichées comme cela par défaut, c'est surtout le cas des éléments de formulaire (comme `<input>`) que nous découvrirons un peu plus tard. Par contre, avec la propriété `display`, nous allons pouvoir transformer d'autres balises en `inline-block`, ce qui va nous aider à réaliser notre design.

Le positionnement inline-block

Les manipulations que demande le positionnement flottant se révèlent parfois un peu délicates sur des sites complexes. Dès qu'il y a un peu plus qu'un simple menu à mettre en page, on risque d'avoir à recourir à des `clear: both;` qui complexifient rapidement le code de la page.

Une meilleure technique consiste à transformer vos éléments en `inline-block` avec la propriété `display`.

Quelques petits rappels sur les éléments de type inline-block :

- Ils se positionnent les uns à côté des autres (exactement ce qu'on veut pour placer notre menu et le corps de notre page !).
- On peut leur donner des dimensions précises (là encore, exactement ce qu'on veut !).

Nous allons transformer en inline-block les deux éléments que nous voulons placer côte à côte : le menu de navigation et la section du centre de la page.

css

```
nav
{
    display: inline-block;
    width: 150px;
    border: 1px solid black;
}

section
{
    display: inline-block;
    border: 1px solid blue;
}
```

Ce qui nous donne comme résultat la figure suivante.

Le menu et le corps sont côte à côte… mais positionnés en bas !

Argh !
Ce n'est pas tout à fait ce qu'on voulait. Et en fait, c'est normal : les éléments inline-block se positionnent sur une même ligne de base (appelée **baseline**), en bas.

Heureusement, le fait d'avoir transformé les éléments en inline-block nous permet d'utiliser une nouvelle propriété, normalement réservée aux tableaux : `vertical-align`. Cette propriété permet de modifier l'alignement vertical des éléments. Voici quelques-unes des valeurs possibles pour cette propriété :

- `baseline` : aligne de la base de l'élément avec celle de l'élément parent (par défaut) ;
- `top` : aligne en haut ;
- `middle` : centre verticalement ;
- `bottom` : aligne en bas ;
- (valeur en px ou %) : aligne à une certaine distance de la ligne de base (baseline).

Il ne nous reste plus qu'à aligner nos éléments en haut (lignes 6 et 13), et le tour est joué (figure suivante) !

css

```
1 nav
2 {
3     display: inline-block;
4     width: 150px;
```

```
    border: 1px solid black;
    vertical-align: top;
}

section
{
    display: inline-block;
    border: 1px solid blue;
    vertical-align: top;
}
```

Le menu et le corps sont alignés en haut et côte à côte

Vous noterez que le corps (la `<section>`) ne prend pas toute la largeur. En effet, ce n'est plus un bloc ! La section occupe seulement la place dont elle a besoin.
Si cela ne vous convient pas pour votre design, modifiez la taille de la section avec `width`.

Et voilà ! Pas besoin de s'embêter avec les marges, aucun risque que le texte passe sous le menu… Bref, c'est parfait !

Les positionnements absolu, fixe et relatif

Il existe d'autres techniques un peu particulières permettant de positionner avec précision des éléments sur la page :

- **Le positionnement absolu** : il nous permet de placer un élément n'importe où sur la page (en haut à gauche, en bas à droite, tout au centre, etc.).
- **Le positionnement fixe** : identique au positionnement absolu mais, cette fois, l'élément reste toujours visible, même si on descend plus bas dans la page. C'est un peu le même principe que `background-attachment: fixed;` (si vous vous en souvenez encore).
- **Le positionnement relatif** : permet de décaler l'élément par rapport à sa position normale.

Comme pour les flottants, les positionnements absolu, fixé et relatif fonctionnent aussi sur des balises de type inline.
Toutefois, vous verrez qu'on l'utilise bien plus souvent sur des balises block que sur des balises inline.

Il faut d'abord faire son choix entre les trois modes de positionnement disponibles. Pour cela, on utilise la propriété CSS `position` à laquelle on donne une de ces valeurs :

- `absolute` : positionnement absolu ;
- `fixed` : positionnement fixe ;
- `relative` : positionnement relatif.

Nous allons étudier un à un chacun de ces positionnements.

Le positionnement absolu

Le positionnement absolu permet de placer un élément (réellement) n'importe où sur la page. Pour effectuer un positionnement absolu, on doit écrire :

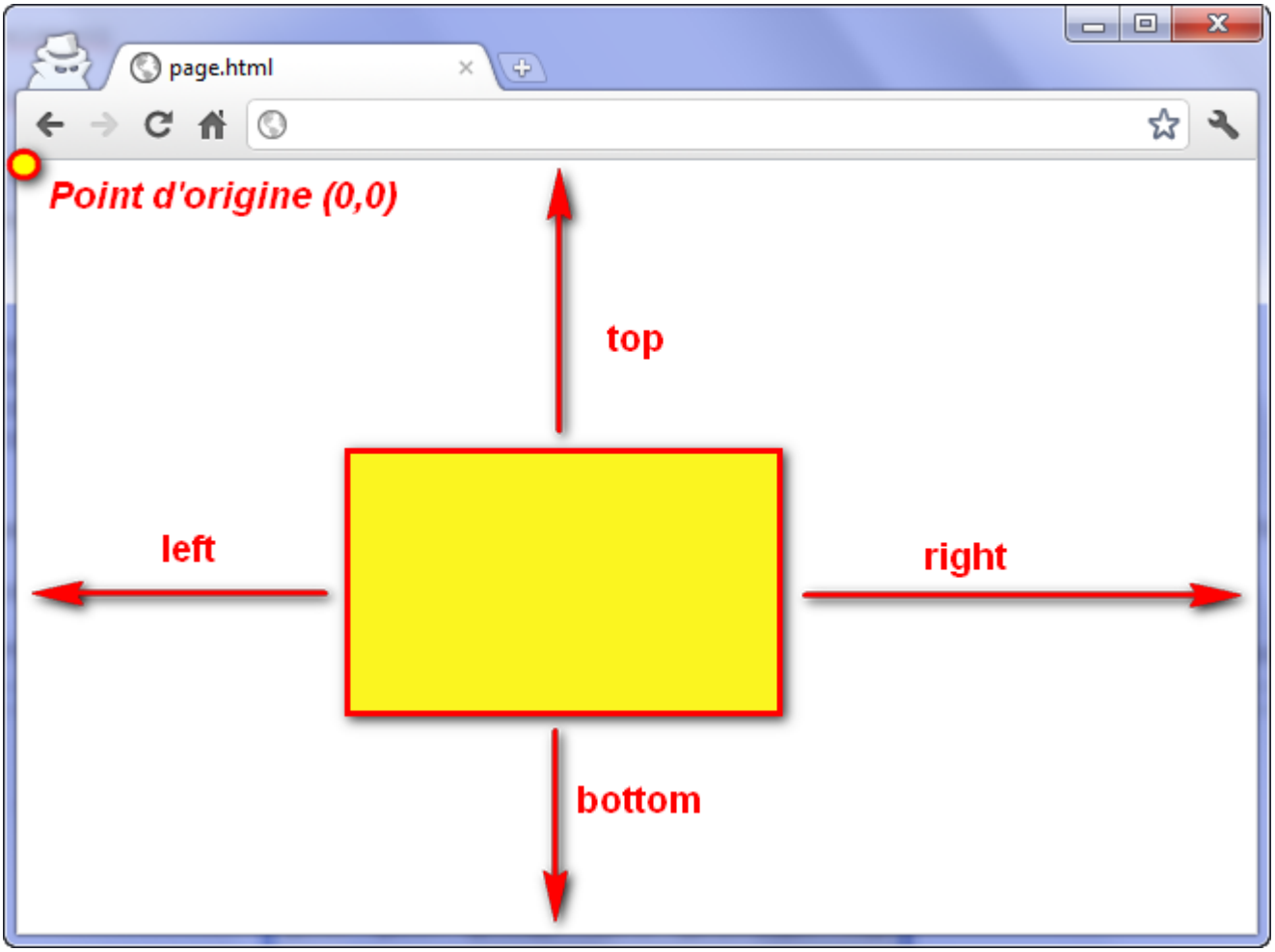
```css
element
{
    position: absolute;
}
```

Mais cela ne suffit pas ! On a dit qu'on voulait un positionnement absolu, mais encore faut-il dire *où l'on veut que le bloc soit positionné sur la page*.
Pour ce faire, on va utiliser quatre propriétés CSS :

- `left` : position par rapport à la gauche de la page ;
- `right` : position par rapport à la droite de la page ;
- `top` : position par rapport au haut de la page ;
- `bottom` : position par rapport au bas de la page.

On peut leur donner une valeur en pixels, comme 14px, ou bien une valeur en pourcentage, comme 50%.

Si ce n'est pas très clair pour certains d'entre vous, la figure suivante devrait vous aider à comprendre.



Positionnement absolu de l'élément sur la page

Avec cela, vous devriez être capables de positionner correctement votre bloc.

Il faut donc utiliser la propriété `position` et au moins une des quatre propriétés ci-dessus (`top` , `left` , `right` ou `bottom`). Si on écrit par exemple :

```css
element
{
    position: absolute;
    right: 0px;
    bottom: 0px;
}
```

… cela signifie que le bloc doit être positionné tout en bas à droite (0 pixel par rapport à la droite de la page, 0 par rapport au bas de la page).

Si on essaye de placer notre bloc `<nav>` en bas à droite de la page, on obtient le même résultat qu'à la figure suivante.

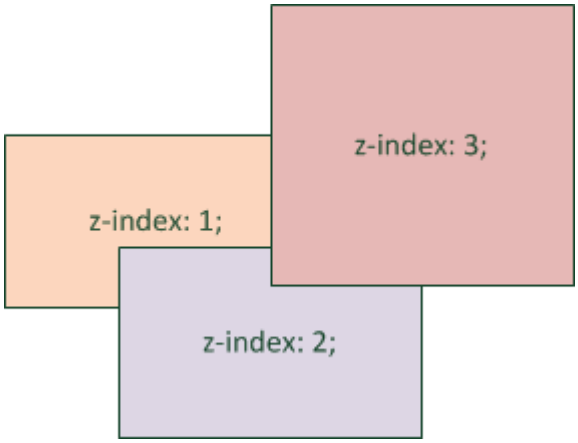
Le menu est positionné en bas à droite de l'écran

On peut bien entendu ajouter une marge intérieure (padding) au menu pour qu'il soit moins collé à sa bordure.

Les éléments positionnés en absolu sont placés par-dessus le reste des éléments de la page ! Par ailleurs, si vous placez deux éléments en absolu vers le même endroit, ils risquent de se chevaucher. Dans ce cas, utilisez la propriété `z-index` pour indiquer quel élément doit apparaître au-dessus des autres.

```css
element
{
    position: absolute;
    right: 0px;
    bottom: 0px;
    z-index: 1;
}
element2
{
    position: absolute;
    right: 30px;
    bottom: 30px;
    z-index: 2;
}
```

L'élément ayant la valeur de `z-index` la plus élevée sera placé par dessus les autres, comme le montre la figure suivante.



Positionnement des éléments absolus

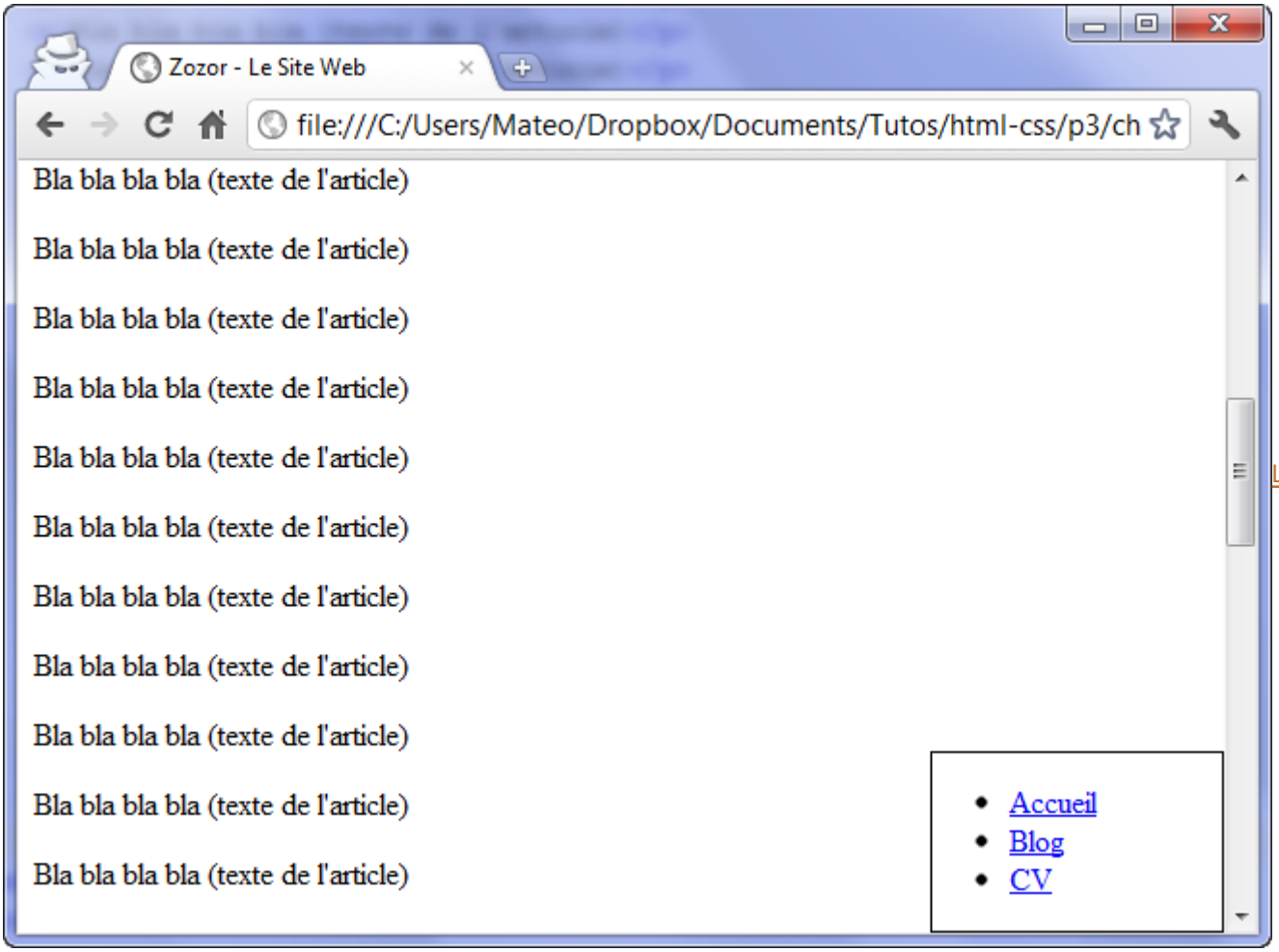
Une petite précision technique qui a son importance : le positionnement absolu ne se fait pas forcément toujours par rapport au coin en haut à gauche de la fenêtre ! Si vous positionnez en absolu un bloc A qui se trouve dans un autre bloc B, lui-même positionné en absolu (ou fixe ou relatif), alors votre bloc A se positionnera par rapport au coin supérieur gauche du bloc B. Faites le test, vous verrez !

Le positionnement fixe

Le principe est *exactement le même* que pour le positionnement absolu sauf que, cette fois, le bloc reste fixe à sa position, même si on descend plus bas dans la page.

```css
element
{
    position: fixed;
    right: 0px;
    bottom: 0px;
}
```

Essayez d'observer le résultat, vous verrez que le menu reste dans le cas présent affiché en bas à droite même si on descend plus bas dans la page (figure suivante).



Le menu reste affiché en bas à droite en toutes circonstances

Le positionnement relatif

Plus délicat, le positionnement relatif peut vite devenir difficile à utiliser. Ce positionnement permet d'effectuer des « ajustements » : l'élément est décalé par rapport à sa position initiale.

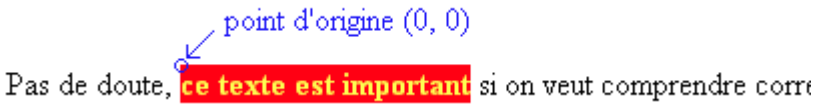
Prenons par exemple un texte important, situé entre deux balises `<strong>`. Pour commencer, je le mets sur fond rouge pour qu'on puisse mieux le repérer :

css

```
strong
{
   background-color: red; /* Fond rouge */
   color: yellow; /* Texte de couleur jaune */
}
```

Cette fois, le schéma que je vous ai montré tout à l'heure pour les positions absolue et fixe ne marche plus. Pourquoi ? Parce que l'origine a changé : le point de coordonnées (0, 0) ne se trouve plus en haut à gauche de votre fenêtre comme c'était le cas tout à l'heure. Non, cette fois l'origine se trouve en haut à gauche… de la position actuelle de votre élément.

Tordu n'est-ce pas ? C'est le principe de la position relative. Le schéma en figure suivante devrait vous aider à comprendre où se trouve l'origine des points.



Origine de la position relative

Donc, si vous faites un `position: relative;` et que vous appliquez une des propriétés `top`, `left`, `right` ou `bottom`, le texte sur fond rouge va se déplacer par rapport à la position où il se trouve.

Prenons un exemple : je veux que mon texte se décale de 55 pixels vers la droite et de 10 pixels vers le bas. Je vais donc demander à ce qu'il soit décalé de 55 pixels par rapport au « bord gauche » et de 10 pixels par rapport au « bord haut » (lignes 6 à 8) :

css

```
1 strong
2 {
3    background-color: red;
4    color: yellow;
5
6    position: relative;
7    left: 55px;
8    top: 10px;
9 }
```

Le texte est alors décalé par rapport à sa position initiale, comme illustré à la figure suivante.

Le texte est décalé

En résumé

- La technique de mise en page la plus récente et la plus puissante est Flexbox. C'est celle que vous devriez utiliser si vous en avez la possibilité.
- D'autres techniques de mise en page restent utilisées, notamment sur des sites plus anciens : le positionnement flottant et le positionnement `inline-block`. Il est conseillé de les connaître.
- Le positionnement flottant (avec la propriété `float`) est l'un des plus utilisés à l'heure actuelle. Il permet par exemple de placer un menu à gauche ou à droite de la page. Néanmoins, cette propriété n'a pas été initialement conçue pour cela et il est préférable, si possible, d'éviter cette technique.
- Le positionnement `inline-block` consiste à affecter un type `inline-block` à nos éléments grâce à la propriété `display`. Ils se comporteront comme des inlines (placement de gauche à droite) mais pourront être redimensionnés comme des blocs (avec `width` et `height`). Cette technique est à préférer au positionnement flottant.
- Le positionnement absolu permet de placer un élément où l'on souhaite sur la page, au pixel près.
- Le positionnement fixe est identique au positionnement absolu mais l'élément restera toujours visible même si on descend plus bas dans la page.

- Le positionnement relatif permet de décaler un bloc par rapport à sa position normale.
- Un élément A positionné en absolu à l'intérieur d'un autre élément B (lui-même positionné en absolu, fixe ou relatif) se positionnera par rapport à l'élément B, et non par rapport au coin en haut à gauche de la page.

☐ J'ai terminé ce chapitre et je passe au suivant

‹ La mise en page avec Flexbox

TP : création d'un site pas à pas ›

L'auteur

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Découvrez aussi ce cours en...

eBook Livre papier PDF Vidéo