

Partie 3 : Mise en page du site

Nous avons appris à construire des pages basiques en HTML, à modifier la mise en forme avec CSS... Intéressons-nous maintenant à la mise en page de notre site ! A la fin de cette partie, nous aboutirons à notre premier site complet, agencé comme nous le voulons !

Structurer sa page

Nous approchons de plus en plus du but. Si nos pages web ne ressemblent pas encore tout à fait aux sites web que nous connaissons, c'est qu'il nous manque les connaissances nécessaires pour faire la mise en page.

En général, une page web est constituée d'un en-tête (tout en haut), de menus de navigation (en haut ou sur les côtés), de différentes sections au centre... et d'un pied de page (tout en bas).

Dans ce chapitre, nous allons nous intéresser aux nouvelles balises HTML dédiées à la structuration du site. Ces balises ont été introduites par HTML5 (elles n'existaient pas avant) et vont nous permettre de dire : « Ceci est mon en-tête », « Ceci est mon menu de navigation », etc.

Pour le moment, nous n'allons pas encore faire de mise en page. Nous allons en fait préparer notre document HTML pour pouvoir découvrir la mise en page dans les prochains chapitres.

Les balises structurantes de HTML5

Je vais vous présenter ici les nouvelles balises introduites par HTML5 pour structurer nos pages. Vous allez voir, cela ne va pas beaucoup changer l'apparence de notre site pour le moment, mais il sera bien construit et prêt à être mis en forme ensuite !

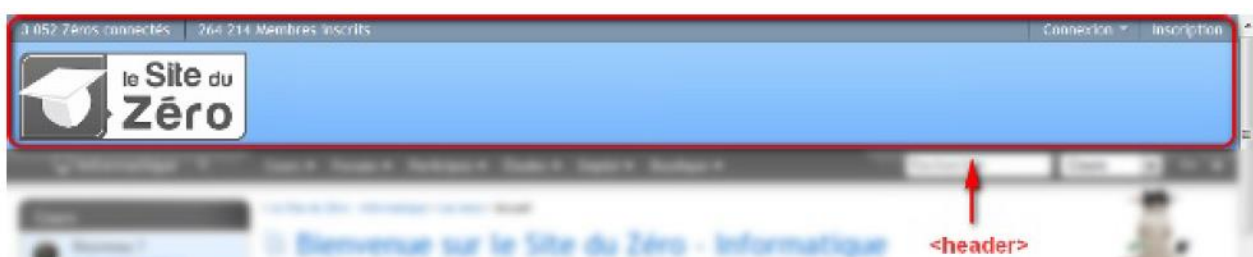
<header> : l'en-tête

La plupart des sites web possèdent en général un en-tête, appelé *header* en anglais. On y trouve le plus souvent un logo, une bannière, le slogan de votre site...

Vous devrez placer ces informations à l'intérieur de la balise **<header>** :

Code : HTML

```
<header>
  <!-- Placez ici le contenu de l'en-tête de votre page -->
</header>
```



L'en-tête peut contenir tout ce que vous voulez : images, liens, textes...

 Il peut y avoir plusieurs en-têtes dans votre page. Si celle-ci est découpée en plusieurs sections, chaque section peut en effet avoir son propre **<header>**.

<footer> : le pied de page

À l'inverse de l'en-tête, le pied de page se trouve en général tout en bas du document. On y trouve des informations comme des liens de contact, le nom de l'auteur, les mentions légales, etc.

Code : HTML

```
<footer>
  <!-- Placez ici le contenu du pied de page -->
</footer>
```



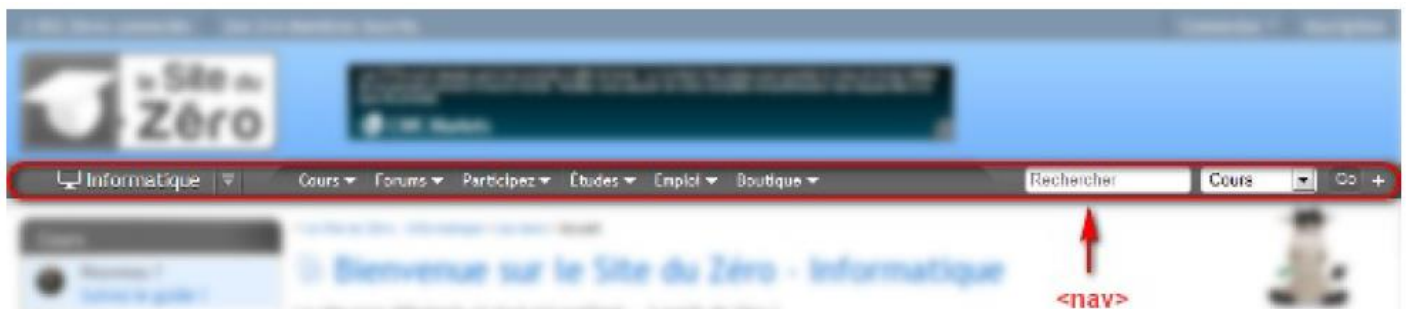
<nav> : principaux liens de navigation

La balise **<nav>** doit regrouper tous les principaux liens de navigation du site. Vous y placerez par exemple le menu principal de votre site.

Généralement, le menu est réalisé sous forme de liste à puces à l'intérieur de la balise **<nav>** :

Code : HTML

```
<nav>
  <ul>
    <li><a href="index.html">Accueil</a></li>
    <li><a href="forum.html">Forum</a></li>
    <li><a href="contact.html">Contact</a></li>
  </ul>
</nav>
```



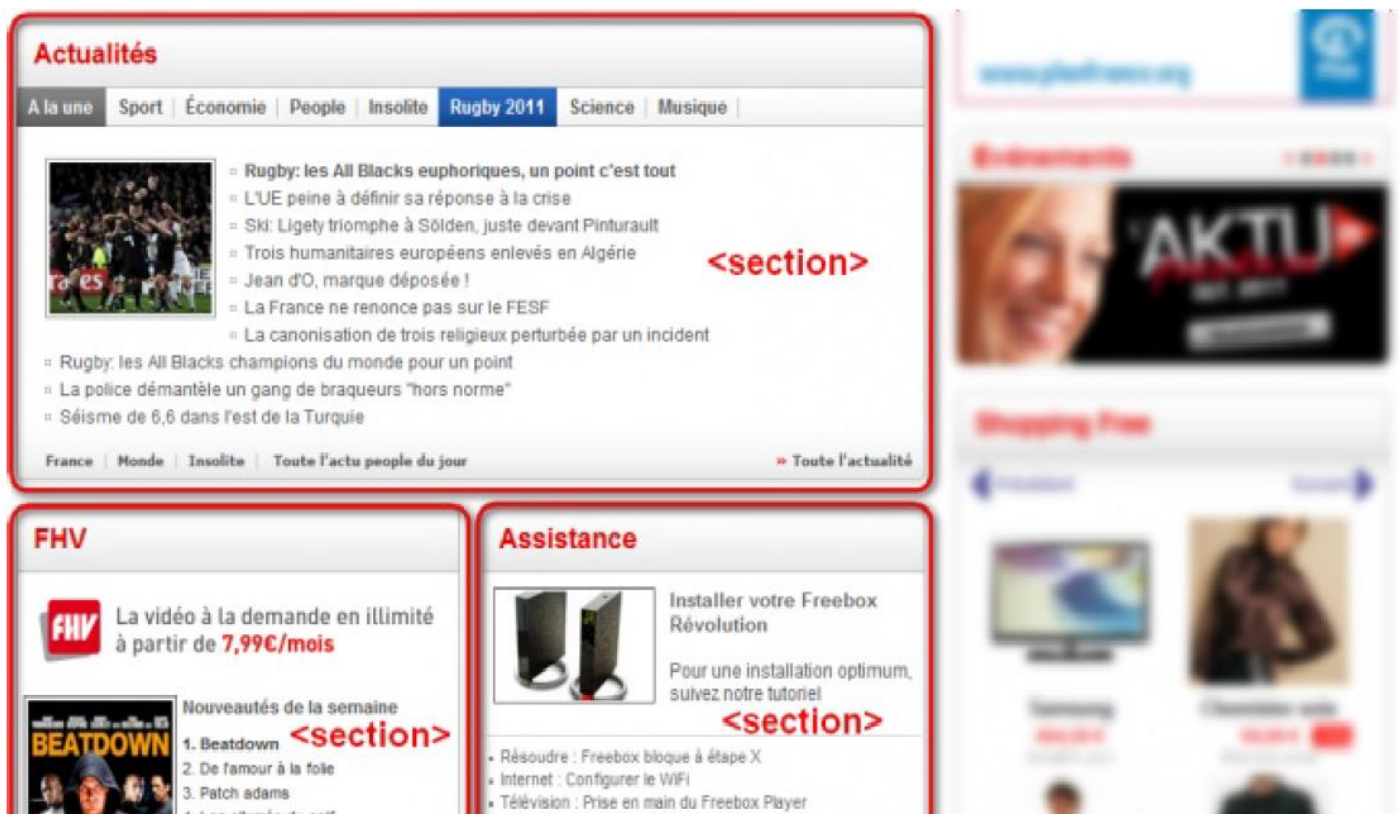
<section> : une section de page

La balise **<section>** sert à regrouper des contenus en fonction de leur thématique. Elle englobe généralement une portion du contenu au centre de la page.

Code : HTML

```
<section>
  <h1>Ma section de page</h1>
  <p>Bla bla bla bla</p>
</section>
```

Sur la page d'accueil du portail Free.fr, on trouve plusieurs blocs qui pourraient être considérés comme des sections de page (figure suivante).



Chaque section peut avoir son titre de niveau 1 (**<h1>**), de même que l'en-tête peut contenir un titre **<h1>** lui aussi. Chacun de ces blocs étant indépendant des autres, il n'est pas illogique de retrouver plusieurs titres **<h1>** dans le code de la page web. On a ainsi « Le titre **<h1>** du **<header>** », « Le titre **<h1>** de cette **<section>** », etc.

<aside> : informations complémentaires

La balise **<aside>** est conçue pour contenir des informations complémentaires au document que l'on visualise. Ces informations sont généralement placées sur le côté (bien que ce ne soit pas une obligation).

Code : HTML

```
<aside>
  <!-- Placez ici des informations complémentaires -->
</aside>
```

Il peut y avoir plusieurs blocs **<aside>** dans la page.

Sur Wikipédia, par exemple, il est courant de voir à droite un bloc d'informations complémentaires à l'article que l'on visualise.

Ainsi, sur la page présentant la planète Saturne (figure suivante), on trouve dans ce bloc les caractéristiques de la planète (dimensions, masse, etc.).



<article> : un article indépendant

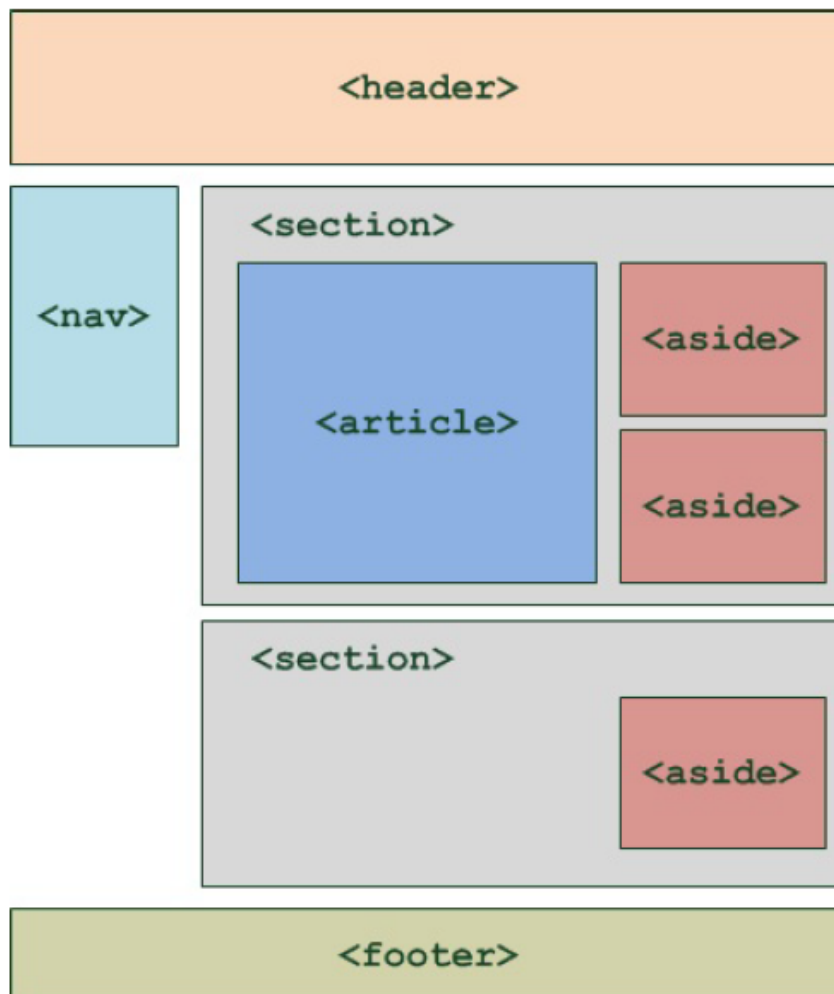
La balise **<article>** sert à englober une portion généralement autonome de la page. C'est une partie de la page qui pourrait ainsi être reprise sur un autre site. C'est le cas par exemple des actualités (articles de journaux ou de blogs).

Code : HTML

```
<article>
  <h1>Mon article</h1>
  <p>Bla bla bla bla</p>
</article>
```




Résumé





que votre menu de navigation soit à droite, ou tout en haut, que vos balises `<aside>` soient au-dessus, etc.

On peut même imaginer une seconde balise `<header>`, placée cette fois à l'intérieur d'une `<section>`. Dans ce cas-là, elle sera considérée comme étant l'en-tête de la section.

Enfin, une section ne doit pas forcément contenir un `<article>` et des `<aside>`. Utilisez ces balises uniquement si vous en avez besoin. Rien ne vous interdit de créer des sections contenant seulement des paragraphes, par exemple.

Exemple concret d'utilisation des balises

Essayons d'utiliser les balises que nous venons de découvrir pour structurer notre page web. Le code ci-dessous reprend toutes les balises que nous venons de voir au sein d'une page web complète :

Code : HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <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>
        <h1>À propos de l'auteur</h1>
        <p>C'est moi, Zozor ! Je suis né un 23 novembre
2005.</p>
      </aside>
      <article>
        <h1>Je suis un grand voyageur</h1>
        <p>Bla bla bla bla (texte de l'article)</p>
      </article>
    </section>
    <footer>
      <p>Copyright Zozor - Tous droits réservés<br />
      <a href="#">Me contacter !</a></p>
    </footer>
  </body>
</html>
```

Ce code peut vous aider à comprendre comment les balises doivent être agencées. Vous y reconnaissez un en-tête, un menu de navigation, un pied de page... et, au centre, une section avec un article et un bloc **<aside>** donnant des informations sur l'auteur de l'article.



À quoi ressemble la page que nous venons de créer ?

À rien !

Si vous testez le résultat, vous verrez juste du texte noir sur fond blanc (figure suivante). C'est normal, il n'y a pas de CSS ! Par contre, la page est bien structurée, ce qui va nous être utile pour la suite.

Zozor

Carnets de voyage

- [Accueil](#)
- [Blog](#)
- [CV](#)

A propos de l'auteur


C'est moi, Zozor ! Je suis né un 23 novembre 2005.


Je suis un grand voyageur

Bla bla bla bla (texte de l'article)

Copyright Zozor - Tous droits réservés

[Me contacter !](#)

 Les liens sont volontairement factices (d'où la présence d'un simple #), ils n'amènent donc nulle part (eh, c'est juste une page de démo) !

 Je ne comprends pas l'intérêt de ces balises. On peut très bien obtenir le même résultat sans les utiliser !

C'est vrai. En fait, ces balises sont seulement là pour expliquer à l'ordinateur « Ceci est l'en-tête », « Ceci est mon pied de page », etc. Elles n'indiquent pas, contrairement à ce qu'on pourrait penser, où doit être placé le contenu. C'est le rôle du CSS, comme nous le verrons dans peu de temps maintenant.

À l'heure actuelle, pour tout vous dire, ces balises ont encore assez peu d'utilité. On pourrait très bien utiliser des balises génériques **<div>** à la place pour englober les différentes portions de notre contenu. D'ailleurs, c'est comme cela qu'on faisait avant l'arrivée de ces nouvelles balises HTML5.

Néanmoins, il est assez probable que, dans un futur proche, les ordinateurs commencent à tirer parti intelligemment de ces nouvelles balises. On peut imaginer par exemple un navigateur qui choisisse d'afficher les liens de navigation **<nav>** de manière toujours visible ! Quand l'ordinateur « comprend » la structure de la page, tout devient possible.

En résumé

Plusieurs balises ont été introduites avec HTML5 pour délimiter les différentes zones qui constituent la page web

<header> : en-tête ;

<footer> : pied de page ;

<nav> : liens principaux de navigation ;

<section> : section de page ;

<aside> : informations complémentaires ;

<article> : article indépendant.

Ces balises peuvent être imbriquées les unes dans les autres. Ainsi, une section peut avoir son propre en-tête.

Ces balises ne s'occupent pas de la mise en page. Elles servent seulement à indiquer à l'ordinateur le sens du texte qu'elles contiennent. On pourrait très bien placer l'en-tête en bas de la page si on le souhaite.

Le modèle des boîtes

Une page web peut être vue comme une succession et un empilement de boîtes, qu'on appelle « blocs ». La plupart des éléments vus au chapitre précédent sont des blocs : `<header>`, `<article>`, `<nav>`... Mais nous connaissons déjà d'autres blocs : les paragraphes `<p>`, les titres `<h1>`...

Dans ce chapitre, nous allons apprendre à manipuler ces blocs comme de véritables boîtes. Nous allons leur donner des dimensions, les agencer en jouant sur leurs marges, mais aussi apprendre à gérer leur contenu... pour éviter que le texte ne dépasse de ces blocs !

Ce sont des notions fondamentales dont nous allons avoir besoin pour mettre en page notre site web... Soyez attentifs !

Les balises de type block et inline

En HTML, la plupart des balises peuvent se ranger dans l'une ou l'autre de deux catégories :

Les balises **inline** : c'est le cas par exemple des liens `<a>`/``.

Les balises **block** : c'est le cas par exemple des paragraphes `<p>`/`</p>`.



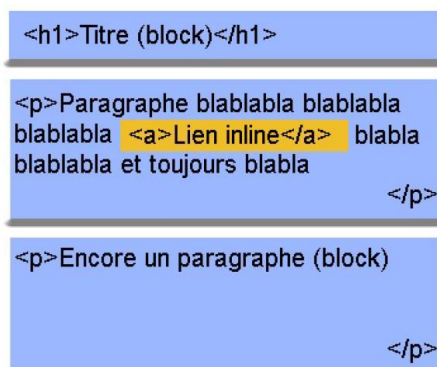
Mais comment je reconnais une balise inline d'une balise block ?

C'est en fait assez facile :

block : une balise de type block sur votre page web crée automatiquement un retour à la ligne avant et après. Il suffit d'imaginer tout simplement un bloc. Votre page web sera en fait constituée d'une série de blocs les uns à la suite des autres. Mais vous verrez qu'en plus, il est possible de mettre un bloc à l'intérieur d'un autre, ce qui va augmenter considérablement nos possibilités pour créer le design de notre site !

inline : une balise de type inline se trouve obligatoirement à l'intérieur d'une balise block. Une balise inline ne crée pas de retour à la ligne, le texte qui se trouve à l'intérieur s'écrit donc à la suite du texte précédent, sur la même ligne (c'est pour cela que l'on parle de balise « en ligne »).

Pour bien visualiser le concept, voici en figure suivante un petit schéma que je vous ai concocté.



- Sur fond bleu, vous avez tout ce qui est de type block.
- Sur fond jaune, vous avez tout ce qui est de type inline.

Comme vous pouvez le voir, les blocs sont les uns en-dessous des autres. On peut aussi les imbriquer les uns à l'intérieur des autres (souvenez-vous, nos blocs `<section>` contiennent par exemple des blocs `<aside>` !).

La balise inline `<a>`, elle, se trouve à l'intérieur d'une balise block et le texte vient s'insérer sur la même ligne.

Quelques exemples

Afin de mieux vous aider à assimiler quelles balises sont inline et quelles balises sont block, voici un petit tableau dressant la liste de quelques balises courantes.

Balises block	Balises inline
<code><p></code>	<code></code>
<code><footer></code>	<code></code>
<code><h1></code>	<code><mark></code>
<code><h2></code>	<code><a></code>
<code><article></code>	<code></code>
...	...

Les balises universelles

Vous les connaissez déjà car je vous les ai présentées il y a quelques chapitres. Ce sont des balises qui n'ont aucun sens particulier (contrairement à `<p>` qui veut dire « paragraphe », `` « important », etc.).

Le principal intérêt de ces balises est que l'on peut leur appliquer une `class` (ou un `id`) pour le CSS quand aucune autre balise ne convient.

Il existe deux balises génériques et, comme par hasard, la seule différence entre les deux est que l'une d'elle est inline et l'autre est block :

`` (inline) ;

`<div></div>` (block).

Les dimensions

Nous allons ici travailler uniquement sur des balises de type block.

Pour commencer, intéressons-nous à la taille des blocs. Contrairement à un inline, un bloc a des dimensions précises. Il possède une largeur et une hauteur. Ce qui fait, ô surprise, qu'on dispose de deux propriétés CSS :

width : c'est la largeur du bloc. À exprimer en pixels (px) ou en pourcentage (%).

height : c'est la hauteur du bloc. Là encore, on l'exprime soit en pixels (px), soit en pourcentage (%).



Pour être exact, **width** et **height** représentent la largeur et la hauteur du contenu des blocs. Si le bloc a des marges (on va découvrir ce principe un peu plus loin), celles-ci s'ajouteront à la largeur et la hauteur.

Par défaut, un bloc prend 100% de la largeur disponible. On peut le vérifier en appliquant à nos blocs des bordures ou une couleur de fond (figure suivante).

Titre

Paragraphe de texte

Maintenant, rajoutons un peu de CSS afin de modifier la largeur des paragraphes. Le CSS suivant dit : « Je veux que tous mes paragraphes aient une largeur de 50% ».

Code : CSS

```
p
{
    width: 50%;
}
```

Le résultat est visible à la figure suivante.

Titre

Paragraphe de texte



Les pourcentages seront utiles pour créer un design qui s'adapte automatiquement à la résolution d'écran du visiteur. Toutefois, il se peut que vous ayez besoin de créer des blocs ayant une dimension précise en pixels :

Code : CSS

```
p
{
    width: 250px;
}
```

Minimum et maximum

On peut demander à ce qu'un bloc ait des dimensions minimales et maximales. C'est très pratique car cela nous permet de définir des dimensions « limites » pour que notre site s'adapte aux différentes résolutions d'écran de nos visiteurs :

min-width : largeur minimale ;

min-height : hauteur minimale ;

max-width : largeur maximale ;

max-height : hauteur maximale.

Par exemple, on peut demander à ce que les paragraphes occupent 50% de la largeur *et* exiger qu'il fassent au moins 400 pixels de large dans tous les cas :

Code : CSS

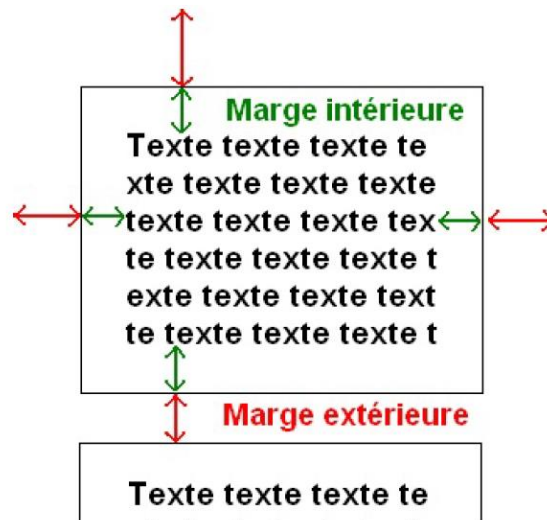
```
p
{
    width: 50%;
    min-width: 400px;
}
```

Observez le résultat en modifiant la largeur de la fenêtre de votre navigateur. Vous allez voir que, si celle-ci est trop petite, le paragraphe se force à occuper au moins 400 pixels de largeur.

Les marges

Il faut savoir que tous les blocs possèdent des marges. Il existe *deux types de marges* : les marges intérieures ; les marges extérieures.

Regardez bien le schéma qui se trouve à la figure suivante.



Sur ce bloc, j'ai mis une bordure pour qu'on repère mieux ses frontières.

- L'espace entre le texte et la bordure est la marge intérieure (en vert).
- L'espace entre la bordure et le bloc suivant est la marge extérieure (en rouge).

En CSS, on peut modifier la taille des marges avec les deux propriétés suivantes :

- **padding** : indique la taille de la marge intérieure. À exprimer en général en pixels (px).
- **margin** : indique la taille de la marge extérieure. Là encore, on utilise le plus souvent des pixels.



Les balises de type *inline* possèdent également des marges. Vous pouvez donc aussi essayer ces manipulations sur ce type de balises.

Pour bien voir les marges, prenons deux paragraphes auxquels j'applique simplement une petite bordure (figure suivante) :

Code : CSS

```
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
}
```


Tests sur les marges

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin dictum ipsum id eros gravida quis bibendum quam suscipit. Pellentesque malesuada posuere justo, id dictum metus tempor et. Nunc luctus lorem vitae justo lacinia sit amet vulputate ligula eleifend. Phasellus porttitor arcu eget elit eleifend vel elementum libero euismod. Sed rhoncus volutpat orci venenatis iaculis. Phasellus faucibus lorem ligula. Integer quis augue in neque luctus consectetur. Proin congue est vitae dolor porttitor tempus. In hac habitasse platea dictumst.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin dictum ipsum id eros gravida quis bibendum quam suscipit. Pellentesque malesuada posuere justo, id dictum

Marges par défaut sur les paragraphes

Comme vous pouvez le constater, il n'y a par défaut pas de marge intérieure (**padding**). En revanche, il y a une marge extérieure (**margin**). C'est cette marge qui fait que deux paragraphes ne sont pas collés et qu'on a l'impression de « sauter une ligne ».

 Les marges par défaut ne sont pas les mêmes pour toutes les balises de type block. Essayez d'appliquer ce CSS à des balises **<div>** qui contiennent du texte, par exemple : vous verrez que, dans ce cas, il n'y a par défaut ni marge intérieure, ni marge extérieure !

Supposons que je veuille rajouter une marge intérieure de 12 px aux paragraphes (figure suivante) :

Code : CSS

```

p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
    padding: 12px; /* Marge intérieure de 12px */
}

```

Tests sur les marges

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin dictum ipsum id eros gravida quis bibendum quam suscipit. Pellentesque malesuada posuere justo, id dictum metus tempor et. Nunc luctus lorem vitae justo lacinia sit amet vulputate ligula eleifend. Phasellus porttitor arcu eget elit eleifend vel elementum libero euismod. Sed rhoncus volutpat orci venenatis iaculis. Phasellus faucibus lorem ligula. Integer quis augue in neque luctus consectetur. Proin congue est vitae dolor porttitor tempus. In hac habitasse platea dictumst.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.



Mais ??? Une marge s'est rajoutée à gauche aussi !

Eh oui, **margin** (comme **padding** d'ailleurs) s'applique aux quatre côtés du bloc.

Si vous voulez spécifier des marges différentes en haut, en bas, à gauche et à droite, il va falloir utiliser des propriétés plus précises... Le principe est le même que pour la propriété **border**, vous allez voir !

En haut, à droite, à gauche, en bas... Et on recommence !

L'idéal serait que vous reteniez les termes suivants en anglais :

- *top* : haut ;
- *bottom* : bas ;
- *left* : gauche ;

- **right** : droite.

Ainsi, vous pouvez retrouver toutes les propriétés de tête.

Je vais quand même vous faire la liste des propriétés pour **margin** et **padding**, histoire que vous soyez sûrs que vous avez compris le principe.

Voici la liste pour **margin** :

- **margin-top** : marge extérieure en haut ;
- **margin-bottom** : marge extérieure en bas ;
- **margin-left** : marge extérieure à gauche ;
- **margin-right** : marge extérieure à droite.

Et la liste pour **padding** :

- **padding-top** : marge intérieure en haut ;
- **padding-bottom** : marge intérieure en bas ;
- **padding-left** : marge intérieure à gauche ;
- **padding-right** : marge intérieure à droite.



Il y a d'autres façons de spécifier les marges avec les propriétés **margin** et **padding**. Par exemple :

margin: 2px 0 3px 1px; signifie « 2 px de marge en haut, 0 px à droite (le px est facultatif dans ce cas), 3 px en bas, 1 px à gauche ».

Autre notation raccourcie : **margin: 2px 1px;** signifie « 2 px de marge en haut et en bas, 1 px de marge à gauche et à droite ».

Centrer des blocs

Il est tout à fait possible de centrer des blocs. C'est même très pratique pour réaliser un design centré quand on ne connaît pas la résolution du visiteur.

Pour centrer, il faut respecter les règles suivantes : donnez une largeur au bloc (avec la propriété **width**) ; indiquez que vous voulez des marges extérieures automatiques, comme ceci : **margin: auto;**.

Essayons cette technique sur nos petits paragraphes (lignes 3 et 4) :

Code : CSS

```
P
{
    width: 350px; /* On a indiqué une largeur (obligatoire) */
    margin: auto; /* On peut donc demander à ce que le bloc soit
centré avec auto */
    border: 1px solid black;
    text-align: justify;
    padding: 12px;
    margin-bottom: 20px;
}
```

Et voici le résultat à la figure suivante.

Centrage

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin dictum ipsum id eros gravida quis bibendum quam suscipit. Pellentesque malesuada posuere justo, id dictum metus tempor et. Nunc luctus lorem vitae justo lacinia sit amet vulputate ligula eleifend. Phasellus porttitor arcu eget elit eleifend vel elementum libero euismod. Sed rhoncus volutpat orci venenatis iaculis. Phasellus faucibus lorem ligula. Integer quis augue in neque luctus consectetur. Proin congue est vitae dolor porttitor tempus. In hac habitasse platea dictumst.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin dictum ipsum id eros gravida quis bibendum quam

Ainsi, le navigateur centre automatiquement nos paragraphes !

 Il n'est cependant pas possible de centrer verticalement un bloc avec cette technique. Seul le centrage horizontal est permis.

Quand ça dépasse...

Lorsqu'on commence à définir des dimensions précises pour nos blocs, comme on vient de le faire, il arrive qu'ils deviennent trop petits pour le texte qu'ils contiennent.

Les propriétés CSS que nous allons voir ici ont justement été créées pour contrôler les dépassements... et décider quoi faire si jamais cela devait arriver.

overflow : couper un bloc

Supposons que vous ayez un long paragraphe et que vous vouliez (pour une raison qui ne regarde que vous) qu'il fasse 250 px de large et 110 px de haut. Ajoutons-lui une bordure et remplissons-le de texte... à ras-bord (figure suivante) :

Code : CSS

```
p
{
    width: 250px;
    height: 110px;
    text-align: justify;
    border: 1px solid black;
}
```

Lorem ipsum dolor sit amet, consectetur
 adipiscing elit. Proin dictum ipsum id eros
 gravida quis bibendum quam suscipit.
 Pellentesque malesuada posuere justo, id
 dictum metus tempor et. Nunc luctus
 lorem vitae justo lacinia sit amet vulputate
 ligula eleifend. Phasellus porttitor arcu
 eget elit eleifend vel elementum libero
 euismod. Sed rhoncus volutpat orci
 venenatis iaculis. Phasellus faucibus lorem
 ligula. Integer quis augue in neque luctus
 consectetur. Proin congue est vitae dolor
 porttitor tempus. In hac habitasse platea
 dictumst.



Horreur ! Le texte dépasse des limites du paragraphe !

Eh oui ! Vous avez demandé des dimensions précises, vous les avez eues ! Mais... le texte ne tient pas à l'intérieur d'un si petit bloc.

Si vous voulez que le texte ne dépasse pas des limites du paragraphe, il va falloir utiliser la propriété **overflow**.

Voici les valeurs qu'elle peut accepter :

- **visible** (par défaut) : si le texte dépasse les limites de taille, il reste visible et sort volontairement du bloc.
- **hidden** : si le texte dépasse les limites, il sera tout simplement coupé. On ne pourra pas voir tout le texte.
- **scroll** : là encore, le texte sera coupé s'il dépasse les limites. Sauf que cette fois, le navigateur mettra en place des barres de défilement pour qu'on puisse lire l'ensemble du texte. C'est un peu comme un cadre à l'intérieur de la page.
- **auto** : c'est le mode « pilote automatique ». En gros, c'est le navigateur qui décide de mettre ou non des barres de défilement (il n'en mettra que si c'est nécessaire). C'est la valeur que je conseille d'utiliser le plus souvent.

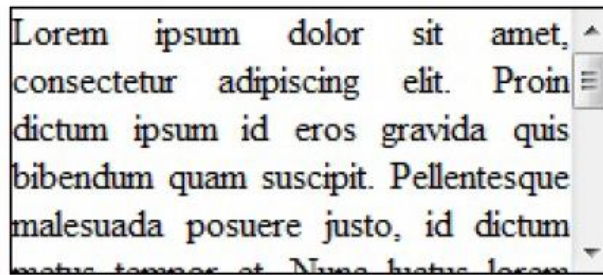
Avec **overflow: hidden**, le texte est donc coupé (on ne peut pas voir la suite), comme sur la figure suivante.

Lorem ipsum dolor sit amet, consectetur
 adipiscing elit. Proin dictum ipsum id eros
 gravida quis bibendum quam suscipit.
 Pellentesque malesuada posuere justo, id
 dictum metus tempor et. Nunc luctus
 lorem vitae justo lacinia sit amet vulputate

Essayons maintenant **overflow: auto**; avec le code CSS suivant (résultat à la figure suivante) :

Code : CSS

```
p
{
    width: 250px;
    height: 110px;
    text-align: justify;
    border: 1px solid black;
    overflow: auto;
}
```



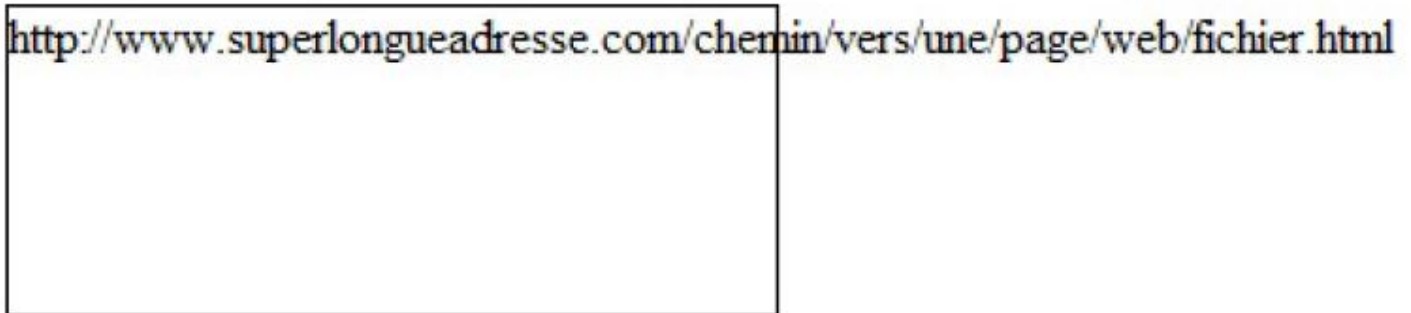
Des barres de défilement nous permettent maintenant de consulter le contenu qui n'était pas visible.

word-wrap : couper les textes trop larges

Si vous devez placer un mot très long dans un bloc, qui ne tient pas dans la largeur, vous allez adorer **word-wrap**.

Cette propriété permet de forcer la césure des très longs mots (généralement des adresses un peu longues).

La figure suivante représente ce que l'on peut avoir quand on écrit une URL un peu longue dans un bloc.



L'ordinateur ne sait pas « couper » l'adresse car il n'y a ni espace, ni tiret. Il ne sait pas faire la césure.

Avec le code suivant, la césure sera forcée dès que le texte risque de dépasser (figure suivante).

Code : CSS

```
p
{
    word-wrap: break-word;
}
```



```
http://www.superlongueadresse.com/chemin/vers/une/page/web/fichier.html
```

En résumé

On distingue deux principaux types de balises en HTML :

Le type block (**<p>**, **<h1>**...) : ces balises créent un retour à la ligne et occupent par défaut toute la largeur disponible. Elles se suivent de haut en bas.

Le type inline (**<a>**, ****...) : ces balises délimitent du texte au milieu d'une ligne. Elles se suivent de gauche à droite.

On peut modifier la taille d'une balise de type block avec les propriétés CSS **width** (largeur) et **height** (hauteur).

On peut définir des minima et maxima autorisés pour la largeur et la hauteur : **min-width**, **max-width**, **minheight**, **max-height**.

Les éléments de la page disposent chacun de marges intérieures (**padding**) et extérieures (**margin**).

S'il y a trop de texte à l'intérieur d'un bloc de dimensions fixes, il y a un risque de débordement. Dans ce cas, il peut être judicieux de rajouter des barres de défilement avec la propriété **overflow** ou de forcer la césure avec **word-wrap**.




Le positionnement en CSS

Voici venu le moment tant attendu : nous allons apprendre à modifier la position des éléments sur notre page.

Vous allez voir, il existe plusieurs techniques permettant d'effectuer la mise en page de son site. Chacune a ses avantages et ses défauts, ce sera à vous de sélectionner celle qui vous semble la meilleure selon votre cas.

Le positionnement flottant

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



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec vitae lorem imperdiet lacus molestie molestie. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec eu purus. Phasellus metus lorem,

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 :

Code : 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>
        <h1>À propos de l'auteur</h1>
        <p>C'est moi, Zozor ! Je suis né un 23 novembre
2005.</p>
      </aside>
      <article>
        <h1>Je suis un grand voyageur</h1>
        <p>Bla bla bla bla (texte de l'article)</p>
      </article>
    </section>
    <footer>
      <p>Copyright Zozor - Tous droits réservés
      <a href="#">Me contacter !</a></p>
    </footer>
  </body>
</html>
```

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

Zozor

Carnets de voyage

- [Accueil](#)
- [Blog](#)
- [CV](#)

A propos de l'auteur

C'est moi, Zozor ! Je suis né un 23 novembre 2005.

Je suis un grand voyageur

Bla bla bla bla (texte de l'article)

Copyright Zozor - Tous droits réservés

[Me contacter !](#)

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 :

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

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

Zozor

Carnets de voyage

<ul style="list-style-type: none"> Accueil Blog CV 	<h3>A propos de l'auteur</h3> <p>C'est moi, Zozor ! Je suis né un 23 novembre 2005.</p>
<h3>Je suis un grand voyageur</h3> <p>Bla bla bla bla (texte de l'article)</p>	

Copyright Zozor - Tous droits réservés

[Me contacter !](#)

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.

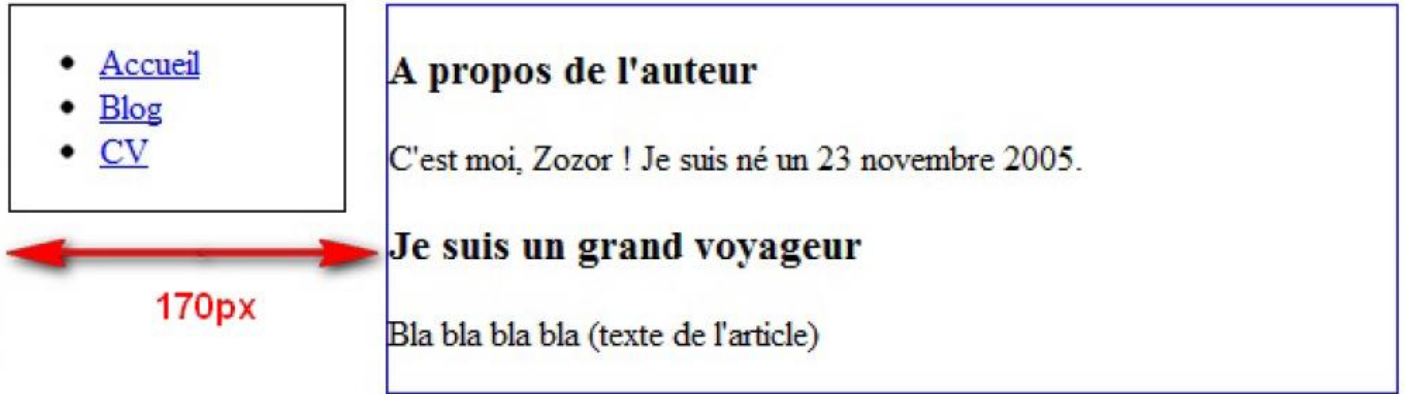
Pour résoudre ces deux problèmes d'un seul coup, il faut ajouter une marge extérieure à gauche de notre **<section>**, marge qui doit être par ailleurs *supérieure* à la largeur du menu. Si notre menu fait 150 px, nous allons par exemple donner une marge extérieure gauche de 170 px à notre section de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

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

section
{
    margin-left: 170px;
    border: 1px solid blue;
}
```

Zozor


Carnets de voyage



Copyright Zozor - Tous droits réservés

[Me contacter !](#)

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 avons déjà découvert, qui oblige la suite du texte à se positionner sous l'élément flottant.

Transformez vos éléments avec display

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 :

```
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	<code><a></code> , <code></code> , <code></code> ...	Éléments d'une ligne. Se placent les uns à côté des autres.
block	<code><p></code> , <code><div></code> , <code><section></code> ...	Éléments en forme de blocs. Se placent les uns en-dessous des autres et peuvent être redimensionnés.
inline-block	<code><select></code> , <code><input></code>	Éléments positionnés les uns à côté des autres (comme les inlines) mais qui peuvent être redimensionnés (comme les blocs).
none	<code><head></code>	Éléments 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 :

Code : 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>`). 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.

Si le positionnement flottant reste, de loin, le mode de positionnement le plus utilisé sur le Web à l'heure actuelle, d'autres techniques existent et bien peu de webmasters le savent. L'une d'elles, étonnamment puissante, est passée sous le nez des concepteurs de sites web alors qu'elle existe depuis CSS 2.1, c'est-à-dire depuis plus de dix ans ! Elle 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.

Code : CSS

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

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

Zozor

Carnets de voyage

<ul style="list-style-type: none">• Accueil• Blog• CV	<h3>A propos de l'auteur</h3> <p>C'est moi, Zozor ! Je suis né un 23 novembre 2005.</p> <h3>Je suis un grand voyageur</h3> <p>Bla bla bla bla (texte de l'article)</p>
---	--

Copyright Zozor - Tous droits réservés

[Me contacter !](#)

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) !

Code : CSS

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

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

Zozor

Carnets de voyage

- [Accueil](#)
- [Blog](#)
- [CV](#)

A propos de l'auteur


C'est moi, Zozor ! Je suis né un 23 novembre 2005.

Je suis un grand voyageur

Bla bla bla bla (texte de l'article)

Copyright Zozor - Tous droits réservés

[Me contacter !](#)


 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`;
- 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 :

Code : 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*.

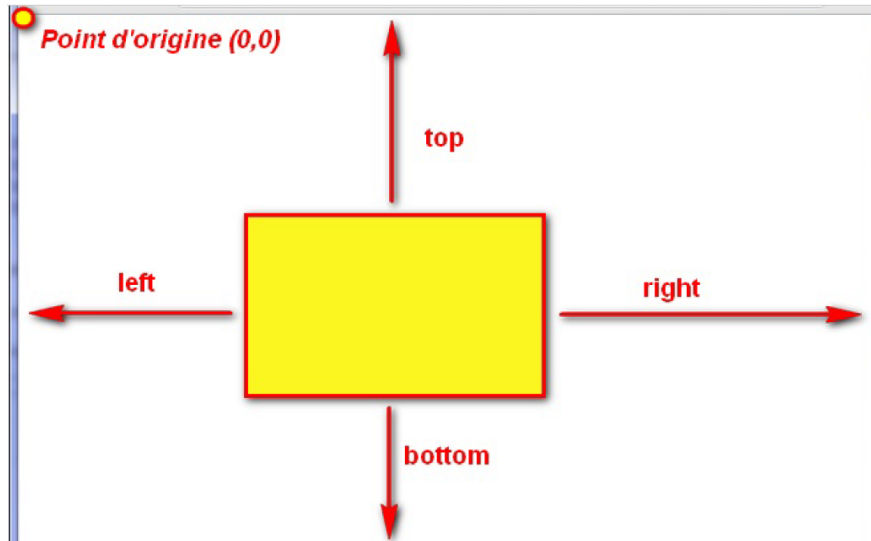
Pour ce faire, on va utiliser quatre propriétés CSS :

- **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.



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 :

Code : 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.

Zozor

Carnets de voyage

A propos de l'auteur

C'est moi, Zozor ! Je suis né un 23 novembre 2005.

Je suis un grand voyageur

Bla bla bla bla (texte de l'article)

Copyright Zozor - Tous droits réservés

[Me contacter !](#)

- [Accueil](#)
- [Blog](#)
- [CV](#)

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.

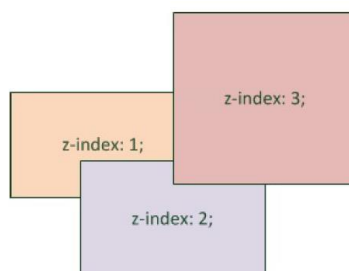
Code : CSS


```

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.



 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.

Code : 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.

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 ****. Pour commencer, je le mets sur fond rouge pour qu'on puisse mieux le repérer :

Code : 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.

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.

Pas de doute,  si on veut comprendre corre

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) :

Code : CSS

```
strong
{
    background-color: red;
    color: yellow;

    position: relative;
    left: 55px;
    top: 10px;
}
```

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



En résumé

- La mise en page d'un site web s'effectue en CSS. Plusieurs techniques sont à notre disposition.
- 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.