

HTML  
CSS



- Dans un répertoire dédié aux exercices , créer une page html nommée tableau.html utilisant les nouveaux éléments cités dans le cours .
- Créer le tableau suivant en utilisant les propriétés de fusion « colspan » et « rowspan »
- Titre planing des tâches par équipes → Text défilant

Équipes	janvier					
	Lundi	Mardi	Mercredi	Jeudi	Vendredi	
Équipe1	tache1			tache2		semaine1
Équipe2	tache1		tache2			
Équipe1	tache3	tache4			tache5	semaine2
Équipe2	tache3		tache4			
Équipe1	tache5			tache6		semaine3
Équipe2	tache5	tache6				
Équipe1	tache7			tache8		semaine4
Équipe2	Tache7		tache8			

Largeur des bordures : 6 pixels ( attribut Cellspacing=)

Épaisseurs de l'ombrage : 2pixels (attribut BORDER=)

Détachement du text par rapport aux bords : 4pixels -> CELLPADDING=

La preniere ligne du tableau avec une couleur d'arriere plan silver

Titre du tableau est EXERCICE02 .

<u>Titre1</u>	<u>Titre2</u>	<u>Titre3</u>
Cellule Simple	Cellule Simple	Cellule Simple
Deux colonnes assemblées		Cellule Simple
Cellule Simple	Cellule Simple	Deux lignes assemblées avec alignement verticle en haut
Cellule Simple	Cellule Simple	
Cellule Simple	Deux lignes assemblées ,deux colonnes assemblées	
Cellule Simple		



il existe une propriété CSS spécifique aux tableaux

## border-collapse

qui signifie « coller les bordures entre elles »  
Cette propriété peut prendre deux valeurs :

**Collapse** : les bordures seront collées entre elles, c'est l'effet qu'on recherche ici

**Separate** : les bordures seront dissociées (valeur par défaut).

Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis



# La ligne d'en-tête

on va rajouter la ligne d'en-tête du tableau.

Dans l'exemple ci-dessous, les en-têtes sont « Nom », « Âge » et « Pays ».

La ligne d'en-tête se crée avec un `<tr>` comme on l'a fait jusqu'ici, mais les cellules qu'elle contient sont, cette fois, encadrées par des

balises `<th>` et non pas `<td>`

```
1 <table>
2   <tr>
3     <th>Nom</th>
4     <th>Âge</th>
5     <th>Pays</th>
6   </tr>
7
8   <tr>
9     <td>Carmen</td>
10    <td>33 ans</td>
11    <td>Espagne</td>
12  </tr>
13  <tr>
14    <td>Michelle</td>
15    <td>26 ans</td>
16    <td>États-Unis</td>
17  </tr>
18 </table>
```

La ligne d'en-tête est très facile à reconnaître pour deux raisons :

- les cellules sont des `<th>` au lieu des `<td>` habituels
- c'est la première ligne du tableau (mais encore faut-il le préciser)

# Titre du tableau

Normalement, tout tableau doit avoir un titre. Le titre permet de renseigner rapidement le visiteur sur le contenu du tableau.

Notre exemple est constitué d'une liste de personnes... oui, mais alors ? Qu'est-ce que cela représente ? Sans titre de tableau, vous le voyez, on est un peu perdu.

Heureusement, il y a **<caption>**

Cette balise se place tout au début du tableau, juste avant l'en-tête. C'est elle qui contient le titre du tableau (figure suivante) :

**Passagers du vol 377**

<b>Nom</b>	<b>Age</b>	<b>Pays</b>
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis



**<table>**

**<caption>Passagers du vol 377</caption>**

**<tr>**

**<th>Nom</th>**

**<th>Âge</th>**

**<th>Pays</th>**

**</tr>**

**<tr>**

**<td>Carmen</td>**

**<td>33 ans</td>**

**<td>Espagne</td>**

**</tr>**

**<tr>**

**<td>Michelle</td>**

**<td>26 ans</td>**

**<td>États-Unis</td>**

**</tr>**

**</table>**



# Un tableau structuré

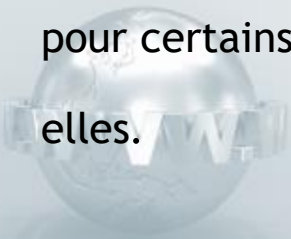
Nous avons appris à construire des petits tableaux simples. Ces petits tableaux suffisent dans la plupart des cas, mais il arrivera que vous ayez besoin de réaliser des tableaux plus... complexes.

Nous allons découvrir deux techniques particulières :

pour les gros tableaux, il est possible de les diviser en trois parties :

- en-tête,
- corps du tableau,
- pied de tableau

pour certains tableaux, il se peut que vous ayez besoin de fusionner des cellules entre elles.





## Diviser un gros tableau

Si votre tableau est assez gros, vous aurez tout intérêt à le découper en plusieurs parties. Pour cela, il existe des balises HTML qui permettent de définir les trois « zones » du tableau :

- l'en-tête (en haut) : il se définit avec les balises `<thead></thead>`
- le corps (au centre) : il se définit avec les balises `<tbody></tbody>`
- le pied du tableau (en bas) : il se définit avec les balises `<tfoot></tfoot>`

Que mettre dans le pied de tableau ? Généralement, si c'est un long tableau, vous y recopiez les cellules d'en-tête. Cela permet de voir, même en bas du tableau, à quoi se rapporte chacune des colonnes. Schématiquement, un tableau en trois parties se découpe donc comme illustré à la figure suivante.

Passagers du vol 377				
En-tête du tableau	Nom	Age	Pays	<thead>
	Carmen	33 ans	Espagne	
Corps du tableau	Michelle	26 ans	Etats-Unis	<tbody>
	François	43 ans	France	
	Martine	34 ans	France	
	Jonathan	13 ans	Australie	
	Xu	19 ans	Chine	
Pied du tableau	Nom	Age	Pays	<tfoot>



```
1 <table>
2   <caption>Passagers du vol 377</caption>
3
4   <thead> <!-- En-tête du tableau -->
5     <tr>
6       <th>Nom</th>
7       <th>Âge</th>
8       <th>Pays</th>
9     </tr>
10  </thead>
11
12  <tfoot> <!-- Pied de tableau -->
13    <tr>
14      <th>Nom</th>
15      <th>Âge</th>
16      <th>Pays</th>
17    </tr>
18  </tfoot>
19
20  <tbody> <!-- Corps du tableau -->
21    <tr>
22      <td>Carmen</td>
23      <td>33 ans</td>
24      <td>Espagne</td>
25    </tr>
26    <tr>
27      <td>Michelle</td>
28      <td>26 ans</td>
29      <td>États-Unis</td>
30    </tr>
```

```
31    <tr>
32      <td>François</td>
33      <td>43 ans</td>
34      <td>France</td>
35    </tr>
36    <tr>
37      <td>Martine</td>
38      <td>34 ans</td>
39      <td>France</td>
40    </tr>
41    <tr>
42      <td>Jonathan</td>
43      <td>13 ans</td>
44      <td>Australie</td>
45    </tr>
46    <tr>
47      <td>Xu</td>
48      <td>19 ans</td>
49      <td>Chine</td>
50    </tr>
51  </tbody>
52 </table>
```

Un tableau s'insère avec la balise **<table>** et se définit ligne par ligne avec **<tr>** .

- Chaque ligne comporte des cellules **<td>** (cellules normales) ou **<th>** (cellules d'en-tête).

- Le titre du tableau se définit avec **<caption>** .

- On peut ajouter une bordure aux cellules du tableau avec **border** .

Pour fusionner les bordures, on utilise la propriété CSS **border-collapse** .

- Un tableau peut être divisé en trois sections : **<thead>** (en-tête), **<tbody>** (corps) et **<tfoot>** (bas du tableau).

L'utilisation de ces balises n'est pas obligatoire.

- On peut fusionner des cellules :

- horizontalement avec l'attribut **colspan**

ou

verticalement avec **rowspan** .

Il faut indiquer combien de cellules doivent être fusionnées.



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).

Donc ,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.



- 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>**

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

La figure suivante, par exemple, représente le site du W3C (qui se charge des nouvelles versions de HTML et CSS, notamment). La partie encadrée en rouge correspondrait à l'en-tête :



L'en-tête du site du W3C

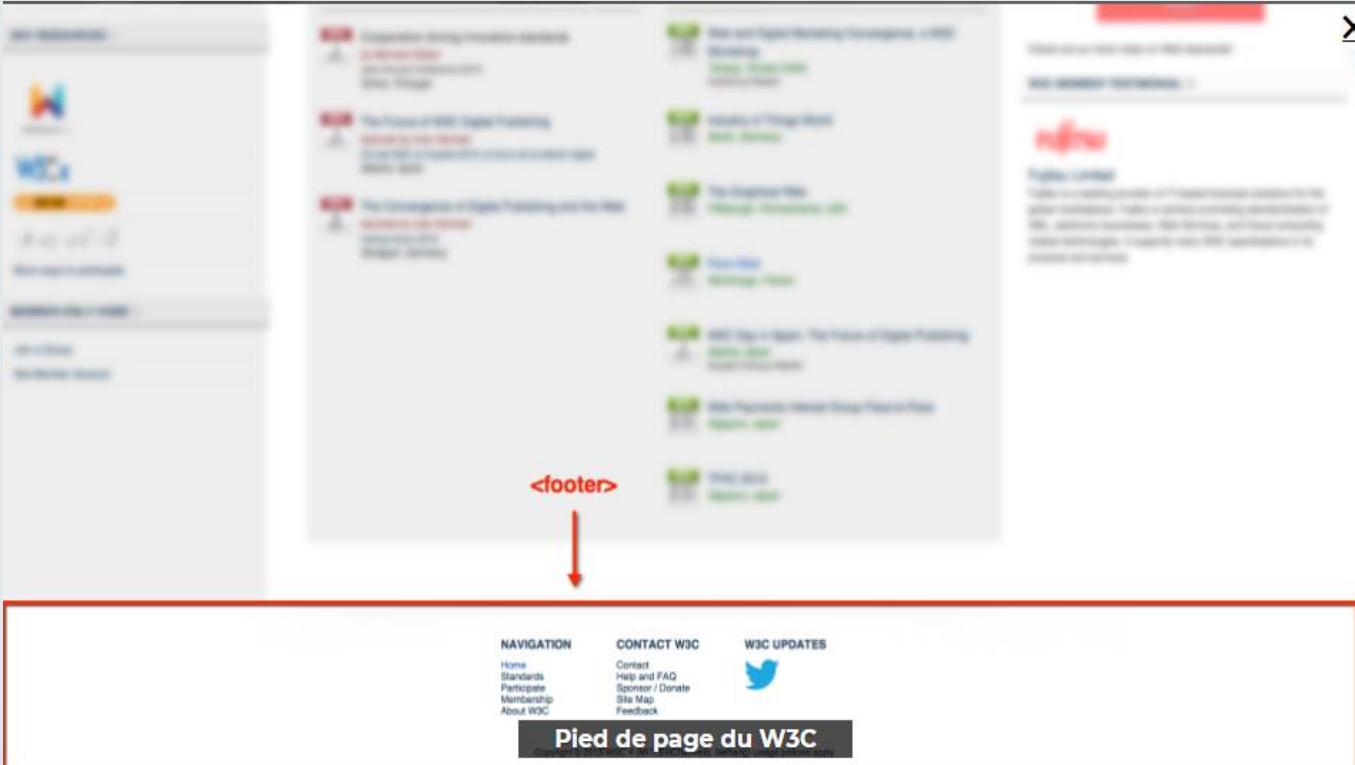
L'en-tête peut contenir tout ce que vous voulez : images, liens, textes...

<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.

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

La figure suivante vous montre à quoi ressemble le pied de page.





## <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>** :



```
1 <nav>
2   <ul>
3     <li><a href="index.html">Accueil</a></li>
4     <li><a href="forum.html">Forum</a></li>
5     <li><a href="contact.html">Contact</a></li>
6   </ul>
7 </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.

```
1 <section>
2   <h1>Ma section de page</h1>
3   <p>Bla bla bla bla</p>
4 </section>
```

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

The screenshot shows a website layout with three distinct sections highlighted by red borders:

- Actualités**: A news section with a navigation bar (A la une, Sport, Économie, People, Insolite, Rugby 2011, Science, Musique) and a list of news items. A red label **<section>** is placed to the right of the list.
- FHV**: A section for video-on-demand services, featuring a list of movies and TV shows. A red label **<section>** is placed to the right of the list.
- Assistance**: A section for technical support, featuring a list of troubleshooting steps and a link to the support site. A red label **<section>** is placed to the right of the list.

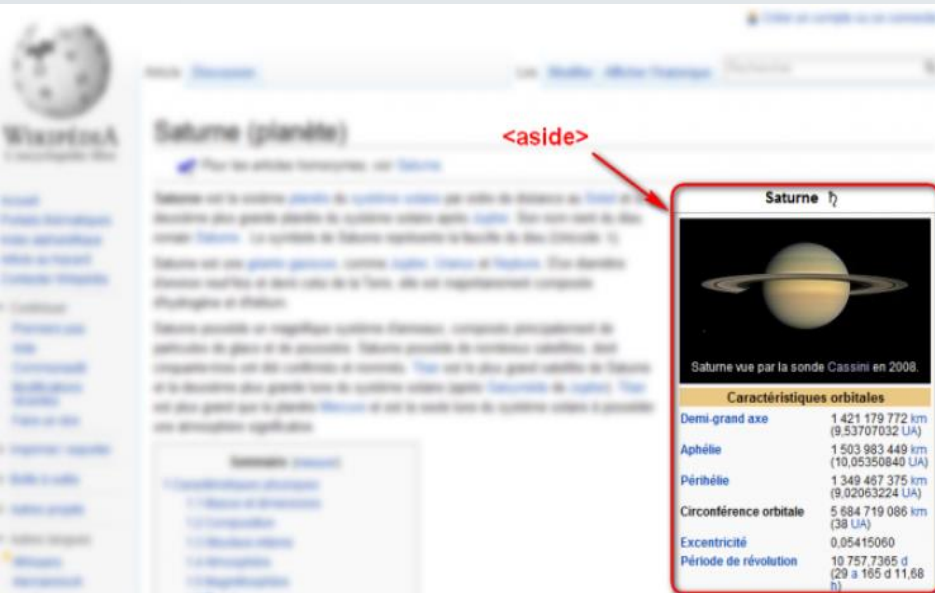
## <aside> : informations complémentaires

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).

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.).

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



## <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.

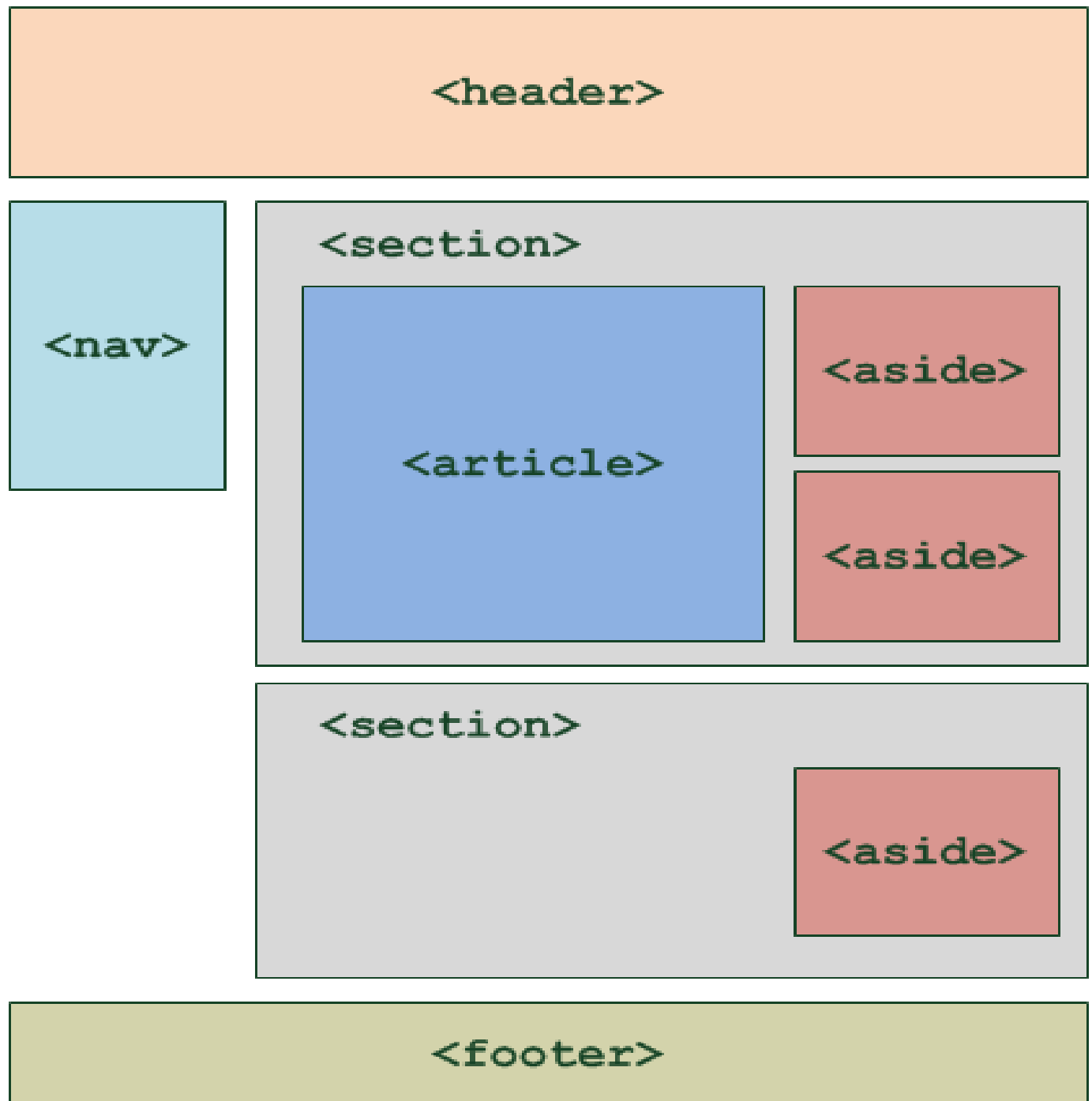
par exemple des actualités (articles de journaux ou de blogs).

```
1 <article>
2   <h1>Mon article</h1>
3   <p>Bla bla bla bla</p>
4 </article>
```

Par exemple, voici un article sur Le Monde :



## Résumé



## Exemple

Essayons d'utiliser les balises que nous venons de découvrir pour structurer notre page web :

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title>Mes sites web </title>
6      </head>
7
8      <body>
9          <header>
10             <h1>salut !</h1>
11             <h2>Carnets de voyage</h2>
12          </header>
13
14          <nav>
15              <ul>
16                  <li><a href="#">Accueil</a></li>
17                  <li><a href="#">Blog</a></li>
18                  <li><a href="#">photos</a></li>
19              </ul>
20          </nav>
21
22          <section>
23              <aside>
24                  <h1>À propos de l'auteur</h1>
25                  <p>éy C'est moi ! ! Je suis né en 23 novembre 2005 .</p>
26              </aside>
27              <article>
28                  <h1>Je suis un grand voyageur</h1>
29                  <p>Bla bla bla bla (texte de l'article)</p>
30              </article>
31          </section>
32
33          <footer>
34              <p>Copyright nom - Tous droits réservés<br />
35              <a href="#">Me contacter !</a></p>
36          </footer>
37
38      </body>
39  </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.



**salut !**

## **Carnets de voyage**

[Accueil](#)

[Blog](#)

[photos](#)

### **À propos de l'auteur**

éy C'est moi !! Je suis né en 23 novembre 2005 .

### **Je suis un grand voyageur**

Bla bla bla bla (texte de l'article)

Copyright nom - Tous droits réservés

[Me contacter !](#)

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 commenceront 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





## Découvrez 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>` ...

Donc , 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 attentif !



## 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></a>`
- les balises **block** : c'est le cas par exemple des paragraphes `<p></p>`

• **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 »).

• Sur fond **bleu**, vous avez tout ce qui est de type block.

• Sur fond **jaune**, vous avez tout ce qui est de type inline.

```
<h1>Titre (block)</h1>
```

```
<p>Paragraphe blablabla blablabla  
blablabla <a>Lien inline</a> blabla  
blablabla et toujours blabla  
</p>
```

```
<p>Encore un paragraphe (block)  
  
</p>
```

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

Quelques exemples ==>



Balises block	Balises inline
<p>	<em>
<footer>	<strong>
<h1>	<mark>
<h2>	<a>
<article>	<img />
...	...

## Les balises universelles

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 :

- `<span></span>` (inline)
- `<div></div>` (block).

Nous allons ici travailler uniquement sur des balises de type block.  
Pour commencer, intéressons-nous à la taille des blocs.

Contrairement à un inline,

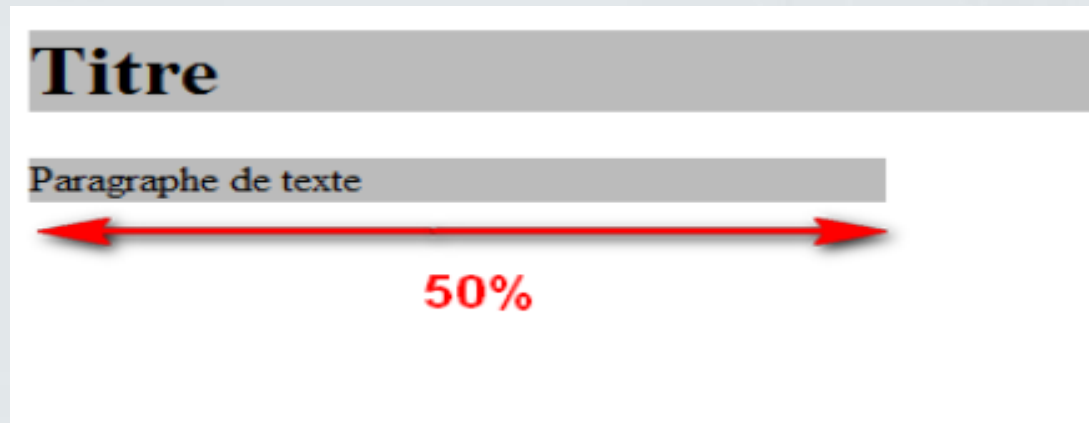
un bloc a des dimensions précises. Il possède une largeur et une hauteur. Ce qui fait, ô surprise, qu'on dispose de deux propriétés CSS :

- **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 (%).

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).



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 % ».



```
1 p
2 {
3     width: 50%;
4 }
```



## 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 :



le paragraphe se force à occuper au moins 400 pixels de largeur.

```
1 p
2 {
3     width: 50%;
4     min-width: 400px;
5 }
```

# Les marges

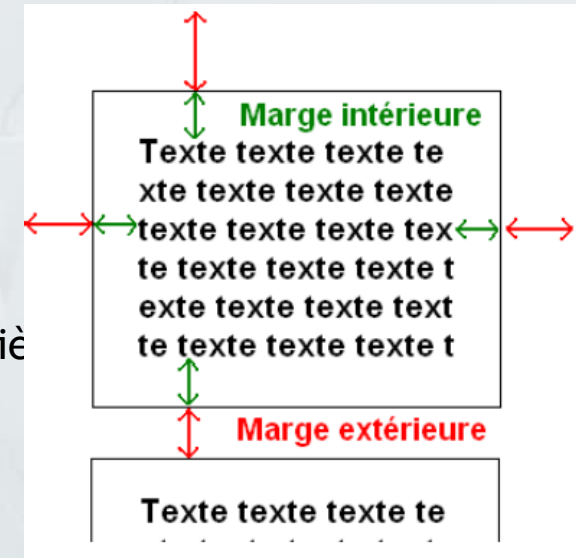
Il faut savoir que tous les blocs possèdent des marges. Il existe deux types de marges :

- les marges intérieures ;
- les marges extérieures.

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.



```
1 p
2 {
3     width: 350px;
4     border: 1px solid black;
5     text-align: justify;
6 }
```

# 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 metus tempor et. Nunc luctus lorem vitae justo lacinia sit amet vulputate ligula eleifend. Phasellus porttitor arcu eget

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 ».

Supposons que je veuille rajouter une marge intérieure de 12 px aux paragraphes :

```
1  p
2  {
3      width: 350px;
4      border: 1px solid black;
5      text-align: justify;
6      padding: 12px; /* Marge intérieure de 12px */
7  }
```





Maintenant, je veux que mes paragraphes soient plus espacés entre eux. Je rajoute la propriété **margin** pour demander à ce qu'il y ait 50 px de marge entre deux paragraphes (figure suivante) :

```
1 p
2 {
3     width: 350px;
4     border: 1px solid black;
5     text-align: justify;
6     padding: 12px;
7     margin: 50px; /* Marge extérieure de 50px */
8 }
```

## 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.

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 `text-align` : **En haut, à droite, en bas, à gauche... 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 sûr d'avoir 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.

## Centrer des blocs

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;`

```
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;
}
```

## 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 suscipit. Pellentesque malesuada posuere justo, id dictum metus tempor et. Nunc luctus lorem vitae justo lacinia sit

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 qu'il fasse 250 px de large et 110 px de haut. Ajoutons-lui une bordure et remplissons-le de texte...

```
1  p
2  {
3      width: 250px;
4      height: 110px;
5      text-align: justify;
6      border: 1px solid black;
7  }
```

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.

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

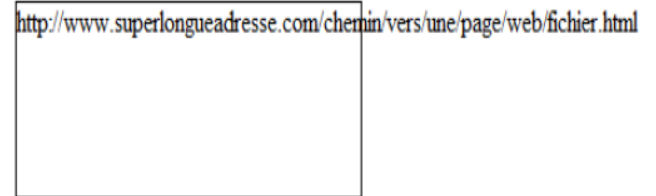
**Auto :** c'est le mode « pilote automatique ». En gros, c'est le navigateur qui décide de mettre ou non des barres de défilement



Si vous devez placer un mot très long dans un bloc, qui ne tient pas dans la largeur .

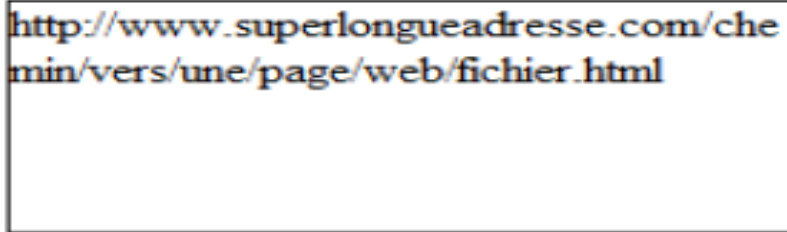
## word-wrap

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.



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

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



http://www.superlongueadresse.com/che  
min/vers/une/page/web/fichier.html

```
1 p
2 {
3     word-wrap: break-word;
4 }
```

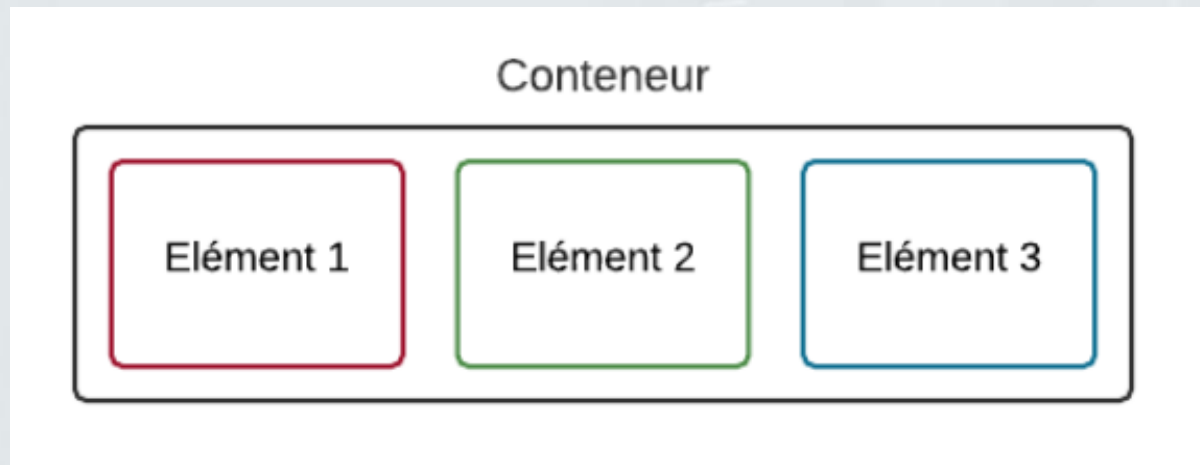


# Faites votre mise en page avec Flexbox


## Un conteneur, des éléments

Le principe de la mise en page avec **Flexbox** est simple : vous définissez un conteneur, et à l'intérieur vous placez plusieurs éléments. Imaginez un carton dans lequel vous rangez plusieurs objets : c'est le principe !

Sur une même page web, vous pouvez sans problème avoir plusieurs conteneurs (plusieurs cartons, si vous préférez) Ce sera à vous d'en créer autant que nécessaire pour obtenir la mise en page que vous voulez.



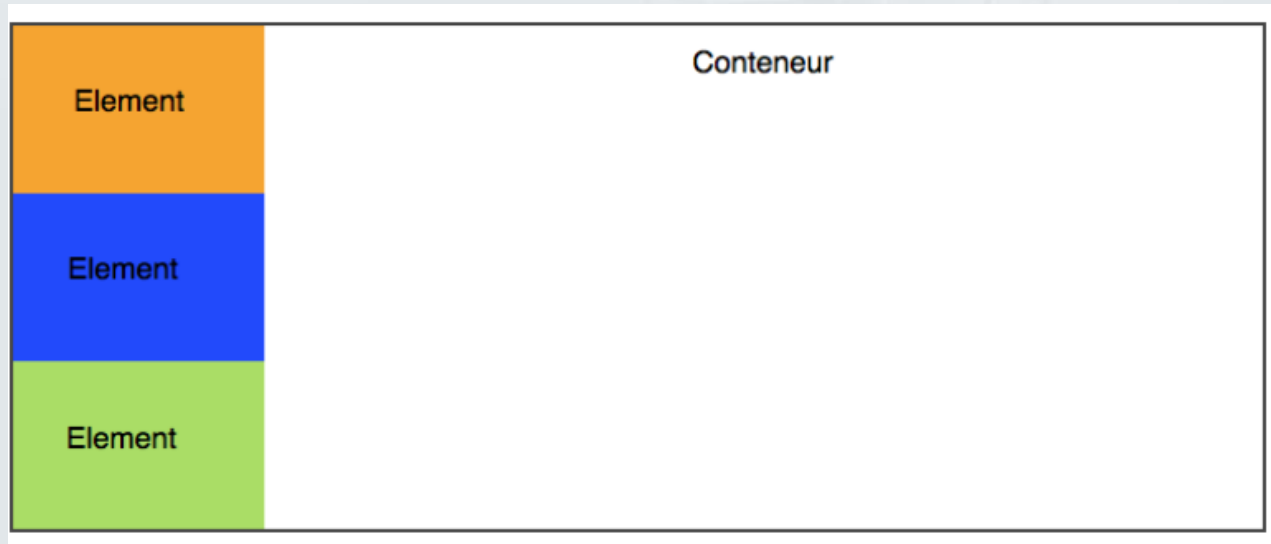
Le conteneur est une balise HTML, et les éléments sont d'autres balises HTML à l'intérieur :



```
1 <div id="conteneur">
2   <div class="element 1">Element </div>
3   <div class="element 2">Element </div>
4   <div class="element 3">Element </div>
5 </div>
```

par défaut, mes éléments vont se mettre les uns en dessous des autres, non ? Ce sont des blocs, après tout !

out à fait. Si je mets une bordure au conteneur, une taille et une couleur de fond aux éléments, on va vite voir comment ils s'organisent :



Pour l'instant, j'ai mis de la couleur sur chacun des éléments, pour que vous puissiez les distinguer facilement.





# Soyez flex !

Découvrons maintenant Flexbox. Si je mets une (une seule !) propriété CSS, tout change. Cette propriété, c'est **flex** et je l'applique au conteneur :

```
1 #conteneur
2 {
3     display: flex;
4 }
```

... alors les blocs se placent par défaut côte à côte.



## La direction

**Flexbox** nous permet d'agencer ces éléments dans le sens que l'on veut. Avec `Flex-direction` on peut les positionner verticalement ou encore les inverser.

=> Il peut prendre les valeurs suivantes :

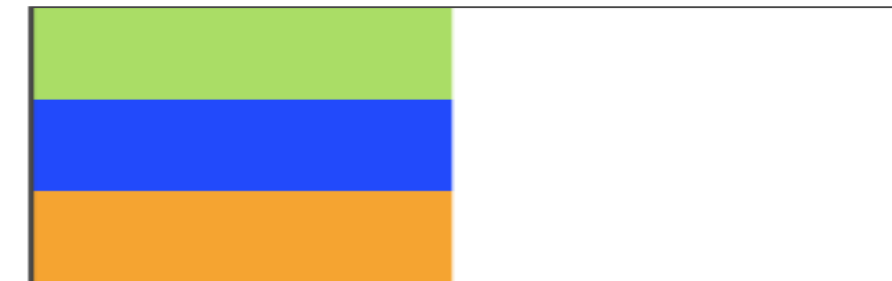
- **row** : organisés sur une ligne (par défaut) ;
- **column** : organisés sur une colonne ;
- **row-reverse** : organisés sur une ligne, mais en ordre inversé ;
- **column-reverse** : organisés sur une colonne, mais en ordre inversé.

### Exemple

```
1 #conteneur
2 {
3     display: flex;
4     flex-direction: column;
5 }
```



```
1 #conteneur
2 {
3     display: flex;
4     flex-direction: column-reverse;
5 }
6
```



## Le retour à la ligne

Par défaut, les blocs essaient de rester sur la même ligne s'ils n'ont pas la place (ce qui peut provoquer des bugs de design, parfois). Si vous voulez, vous pouvez demander à ce que les blocs aillent à la ligne lorsqu'ils n'ont plus la place, avec **flex-wrap** qui peut prendre ces valeurs :

- **nowrap** : pas de retour à la ligne (par défaut) .
- **wrap** : les éléments vont à la ligne lorsqu'il n'y a plus la place .
- **wrap-reverse** : les éléments vont à la ligne, lorsqu'il n'y a plus la place, en sens inverse.

```
1 #conteneur
2 {
3     display: flex;
4     flex-wrap: wrap;
5 }
6
```

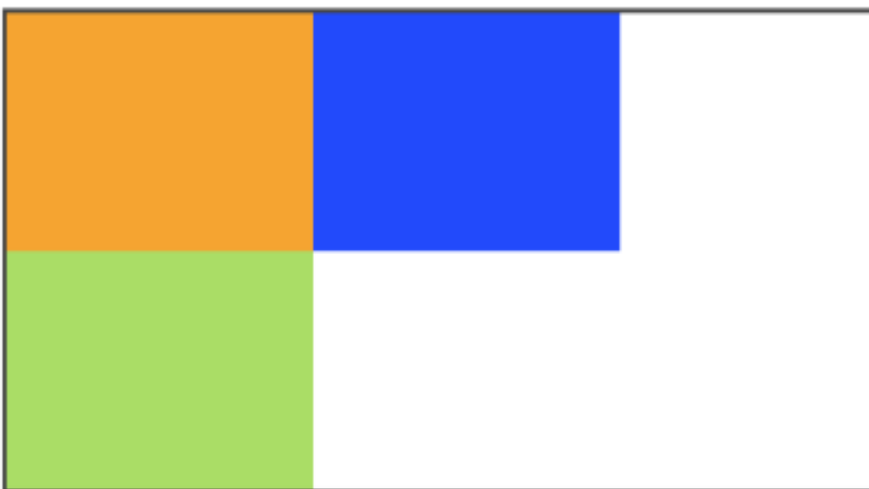


nowrap



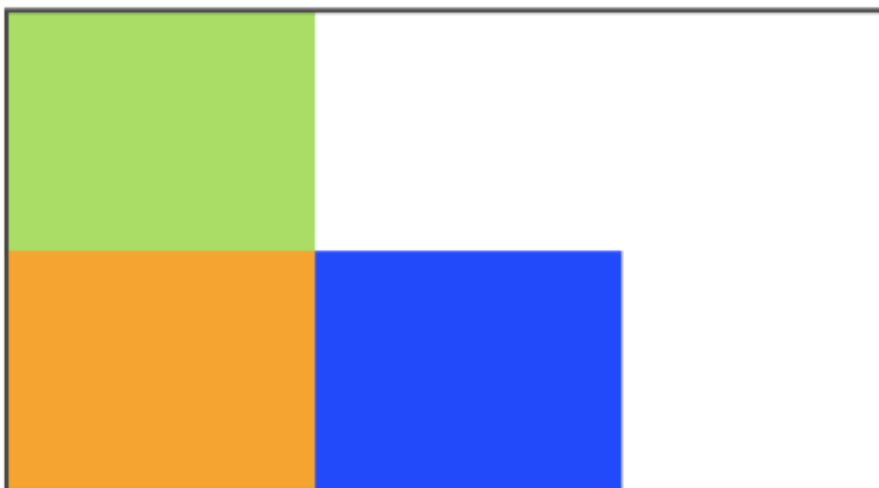
Les éléments se resserrent tant qu'ils peuvent

wrap



Les éléments passent à la ligne

wrap-reverse



Les éléments passent à la ligne à l'envers

## Alignez-les !

Les éléments sont organisés soit horizontalement (par défaut), soit verticalement. Cela définit ce qu'on appelle l'axe principal. Il y a aussi un axe secondaire (cross axis) :

- si vos éléments sont organisés horizontalement, l'axe secondaire est l'axe vertical ;
- si vos éléments sont organisés verticalement, l'axe secondaire est l'axe horizontal.

Pourquoi je vous raconte ça ? Parce que nous allons découvrir comment aligner nos éléments sur l'axe principal et sur l'axe secondaire.

### Aligner sur l'axe principal

Pour faire simple, partons sur des éléments organisés horizontalement (c'est le cas par défaut).

Pour changer leur alignement, on va utiliser **justify-content** qui peut prendre ces valeurs :

- **flex-start** : alignés au début (par défaut) .
- **flex-end** : alignés à la fin .
- **center** : alignés au centre .
- **space-between** : les éléments sont étirés sur tout l'axe (il y a de l'espace entre eux) .
- **space-around** : idem, les éléments sont étirés sur tout l'axe, mais ils laissent aussi de l'espace sur les extrémités.

Par exemple :

```
1 #conteneur
2 {
3     display: flex;
4     justify-content: space-around;
5 }
```



flex-start



flex-end



center



space-between



space-around

ça marche aussi si vos éléments sont dans une direction verticale.  
Dans ce cas, l'axe vertical devient l'axe principal, et **justify-content** s'applique aussi :

```
1 #conteneur
2 {
3     display: flex;
4     flex-direction: column;
5     justify-content: center;
6     height: 350px; /* Un peu de hauteur pour que les éléments aient la place de bouger */
7 }
```



## Aligner sur l'axe secondaire

Comme je vous le disais, si nos éléments sont placés dans une direction horizontale (ligne), l'axe secondaire est... vertical. Et inversement : si nos éléments sont dans une direction verticale (colonne), l'axe secondaire est horizontal. Avec **align-items** nous pouvons changer leur alignement sur l'axe secondaire. Il peut prendre ces valeurs :

- **stretch** : les éléments sont étirés sur tout l'axe (valeur par défaut) .
- **flex-start** : alignés au début .
- **flex-end** : alignés à la fin .
- **center** : alignés au centre .
- **baseline** : alignés sur la ligne de base (semblable à flex-start).

Pour ces exemples, nous allons partir du principe que nos éléments sont dans une direction horizontale.

```
1 #conteneur
2 {
3   display: flex;
4   justify-content: center;
5   align-items: center;
6 }
```



Un alignement sur l'axe secondaire avec align-items nous permet de centrer complètement l'élément dans le conteneur !

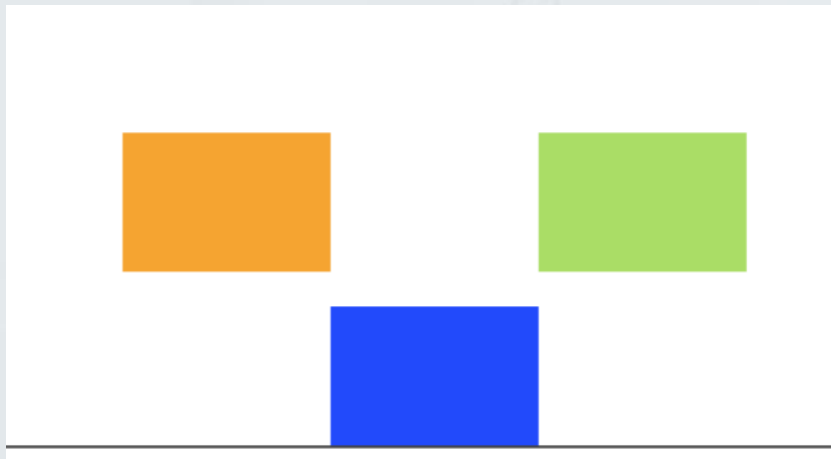




## Aligner un seul élément

il est possible de faire une exception pour un seul des éléments sur l'axe secondaire avec **align-self**

```
1 #conteneur
2 {
3   display: flex;
4   flex-direction: row;
5   justify-content: center;
6   align-items: center;
7 }
8
9 .element:nth-child(2) /* On prend le deuxième bloc élément */
10 {
11   background-color: blue;
12   align-self: flex-end; /* Seul ce bloc sera aligné à la fin */
13 }
```



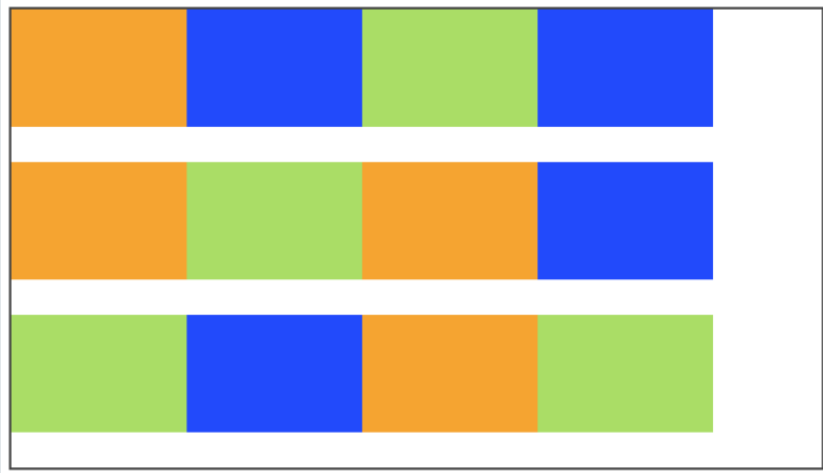
Si vous avez plusieurs lignes dans votre Flexbox, vous pouvez choisir comment celles-ci seront réparties avec **align-content**

Prenons donc un cas de figure où nous avons plusieurs lignes. Je vais rajouter des éléments :

```
1 <div id="conteneur">
2   <div class="element"></div>
3   <div class="element"></div>
4   <div class="element"></div>
5   <div class="element"></div>
6   <div class="element"></div>
7   <div class="element"></div>
8   <div class="element"></div>
9   <div class="element"></div>
10  <div class="element"></div>
11  <div class="element"></div>
12  <div class="element"></div>
13  <div class="element"></div>
14 </div>
```



J'autorise mes éléments à aller à la ligne avec **flex-wrap**

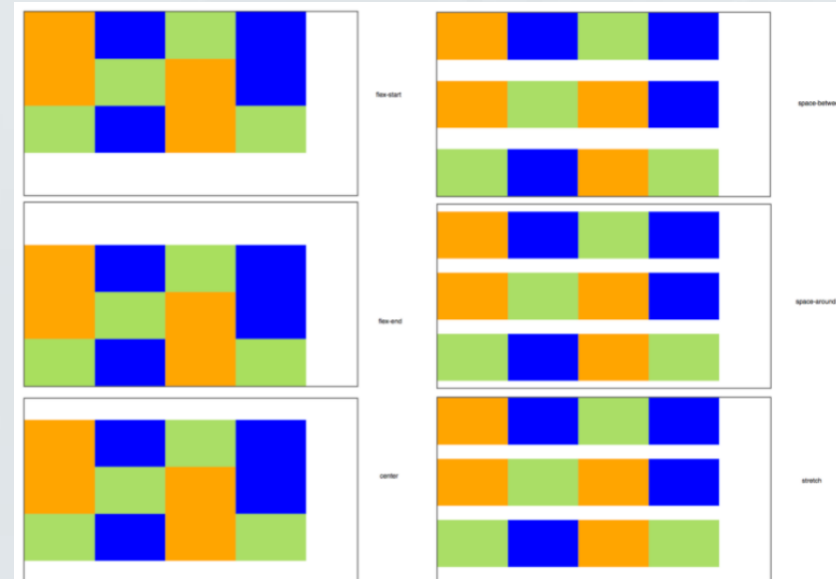


```
1 #conteneur
2 {
3     display: flex;
4     flex-wrap: wrap;
5 }
```

Jusque là, rien de vraiment nouveau. Voyons voir comment les lignes se répartissent différemment avec la nouvelle propriété **align-content** que je voulais vous présenter. Elle peut prendre ces valeurs :

- flex-start : les éléments sont placés au début .
- flex-end : les éléments sont placés à la fin .
- center : les éléments sont placés au centre .
- space-between : les éléments sont séparés avec de l'espace entre eux .
- space-around : il y a de l'espace au début et à la fin .
- stretch (par défaut) : les éléments s'étirent pour occuper tout l'espace.

Voici ce que donnent les différentes valeurs :



## Rappel à l'ordre

Sans changer le code HTML, nous pouvons modifier l'ordre des éléments en CSS grâce à la propriété **order**. Indiquez simplement un nombre, et les éléments seront triés du plus petit au plus grand nombre.

Reprenons une simple ligne de 3 éléments :

```
1 #conteneur
2 {
3     display: flex;
4 }
```



Si je dis que le premier élément sera placé en 3e position, le second en 1re position et le troisième en 2de position, l'ordre à l'é

```
1 .element:nth-child(1)
2 {
3     order: 3;
4 }
5 .element:nth-child(2)
6 {
7     order: 1;
8 }
9 .element:nth-child(3)
10 {
11     order: 2;
12 }
```



Avec la propriété flex , nous pouvons permettre à un élément de grossir pour occuper tout l'espace restant.

```
1 .element:nth-child(2)
2 {
3     flex: 1;
4 }
```



Ici, le premier élément peut grossir 2 fois plus que le deuxième élément :

```
1  .element:nth-child(1)
2  {
3      flex: 2;
4  }
5  .element:nth-child(2)
6  {
7      flex: 1;
8  }
```



Vous allez effectuer la mise en page d'un site qui s'appelle 'Le blog trotter' ! Pour cela, effectuez les tâches suivantes :

1. Une balise sémantique `<nav>` manque, ajoutez-la au bon endroit.
2. Retirez les puces de la liste à puces (à vous de trouver comment faire !).
3. Placez le header et le menu côte à côte.
4. Affichez les paragraphes en justifié, sur 80 % de largeur, et centrez leurs blocs sur la page.

## Le blog trotter

[Accueil](#)  
[Archives](#)  
[Contact](#)

Je parcours la planète... et vous la fais découvrir !

### La Chine

On peut aujourd'hui aller de Paris à Pékin en 10h d'avion. Si la Chine semble de moins en moins lointaine, son pouvoir d'attraction et son mystère demeurent. Depuis son ouverture économique commencée en 1992, la voilà devenue enfin accessible aux voyageurs et aux routards. Pour aller en Chine, un simple visa suffit.

Pas facile de découvrir en 3 semaines un pays vaste comme 17 fois la France. Au nord et à Pékin, la Grande Muraille, la Cité interdite, le palais d'Été, les musées et les temples. Au centre du pays, l'Armée enterrée de Xi'an. Non loin de la mer, Shanghai, ville symbole du nouveau capitalisme chinois. Enfin, c'est une autre Chine, raffinée, secrète et poétique, qui se cache dans les jardins de Suzhou ou sur les berges de la rivière Li, dans le Sud, près de Guilin ou encore le long des rizières et plantations de thé du Yunnan. Source : [Le Routard](#)

### L'Espagne

Diversité des paysages, des cultures, des langues (castillan, catalan, basque), des terroirs et des villes. L'Espagne s'offre à tous les goûts : laissons de côté les plages envahies l'été et la Costa del Sol bétonnée par les complexes hôteliers. Aventureons-nous plutôt dans l'intérieur du pays, superbe et naturel, prodigue en paysages saisissants, en monuments splendides, en modes de vie passionnants...

Pour cela, il suffit parfois de s'éloigner d'une dizaine de kilomètres des foules. Découvrir Salamanque, la petite Rome espagnole, ou Tolède la belle médiévale perchée sur son promontoire. Parcourir le rude plateau de Castille, de Ségovie à Léon, à la découverte de magnifiques cathédrales et d'extraordinaires musées. Goûter à la spécificité de l'âme catalane et à ses trésors artistiques, de Dalí à Gaudí en passant par Tàpiès et Miró. Prendre le chemin de Saint-Jacques-de-Compostelle pour admirer les beautés de la rurale Galice, aussi verdoyante que la terre des Basques, à laquelle ils sont tant attachés. Ou, enfin, un soir de fête à Séville, Madrid ou Barcelone, succomber à cet éclatant bonheur de vivre qui est la marque du pays tout entier. Car les nuits y sont souvent plus intenses que les jours. Source : [Le Routard](#)

Copyright Le Blog Trotter



## Le positionnement flottant

Reprenons le code HTML structuré que nous avons déjà réalisé :

**salut !**

### **Carnets de voyage**

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

### **À propos de l'auteur**

éy C'est moi ! ! Je suis né en 23 novembre 2005 .

### **Je suis un grand voyageur**

Bla bla bla bla (texte de l'article)

Copyright nom - 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 :

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





# salut !

## Carnets de voyage

[Accueil](#)  
[Blog](#)  
[photos](#)

### À propos de l'auteur

éy C'est moi ! ! Je suis né en 23 novembre 2005 .

### Je suis un grand voyageur

Bla bla bla bla (texte de l'article)

Copyright nom - Tous droits réservés

[Me contacter !](#)

Il y a deux défauts :

- 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 nom») 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).

```
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 }
```

# salut !

## Carnets de voyage

[Accueil](#)  
[Blog](#)  
[photos](#)

### À propos de l'auteur

éy C'est moi !! Je suis né en 23 novembre 2005 .

### Je suis un grand voyageur

Bla bla bla bla (texte de l'article)

## 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, je peux par exemple imposer à mes liens (originellement de type inline) d'apparaître sous forme de blocs :

```
1 a
2 {
3     display: block;
4 }
```

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 !

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



## Quelques petits rappels sur les éléments de **type inline-block**

- ils se positionnent les uns à côté des autres.
- on peut leur donner des dimensions précises .
- 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.

```
1 nav
2 {
3     display: inline-block;
4     width: 150px;
5     border: 1px solid black;
6 }
7
8 section
9 {
10    display: inline-block;
11    border: 1px solid blue;
12 }
```

**salut !**

**Carnets de voyage**

**À propos de l'auteur**

éy C'est moi ! ! Je suis né en 23 novembre 2005 .

**Je suis un grand voyageur**

Bla bla bla bla (texte de l'article)

[Accueil](#)  
[Blog](#)  
[photos](#)

Copyright nom - 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.

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 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 .



```
1  nav
2  {
3      display: inline-block;
4      width: 150px;
5      border: 1px solid black;
6      vertical-align: top;
7  }
8
9  section
10 {
11     display: inline-block;
12     border: 1px solid blue;
13     vertical-align: top;
14 }
```

**salut !**

## Carnets de voyage

[Accueil](#)  
[Blog](#)  
[photos](#)

### À propos de l'auteur

éy C'est moi ! ! Je suis né en 23 novembre 2005 .

### Je suis un grand voyageur

Bla bla bla bla (texte de l'article)

Copyright nom - Tous droits réservés

[Me contacter !](#)



## 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.

## 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 :

```
1 element
2 {
3     position: absolute;
4 }
```

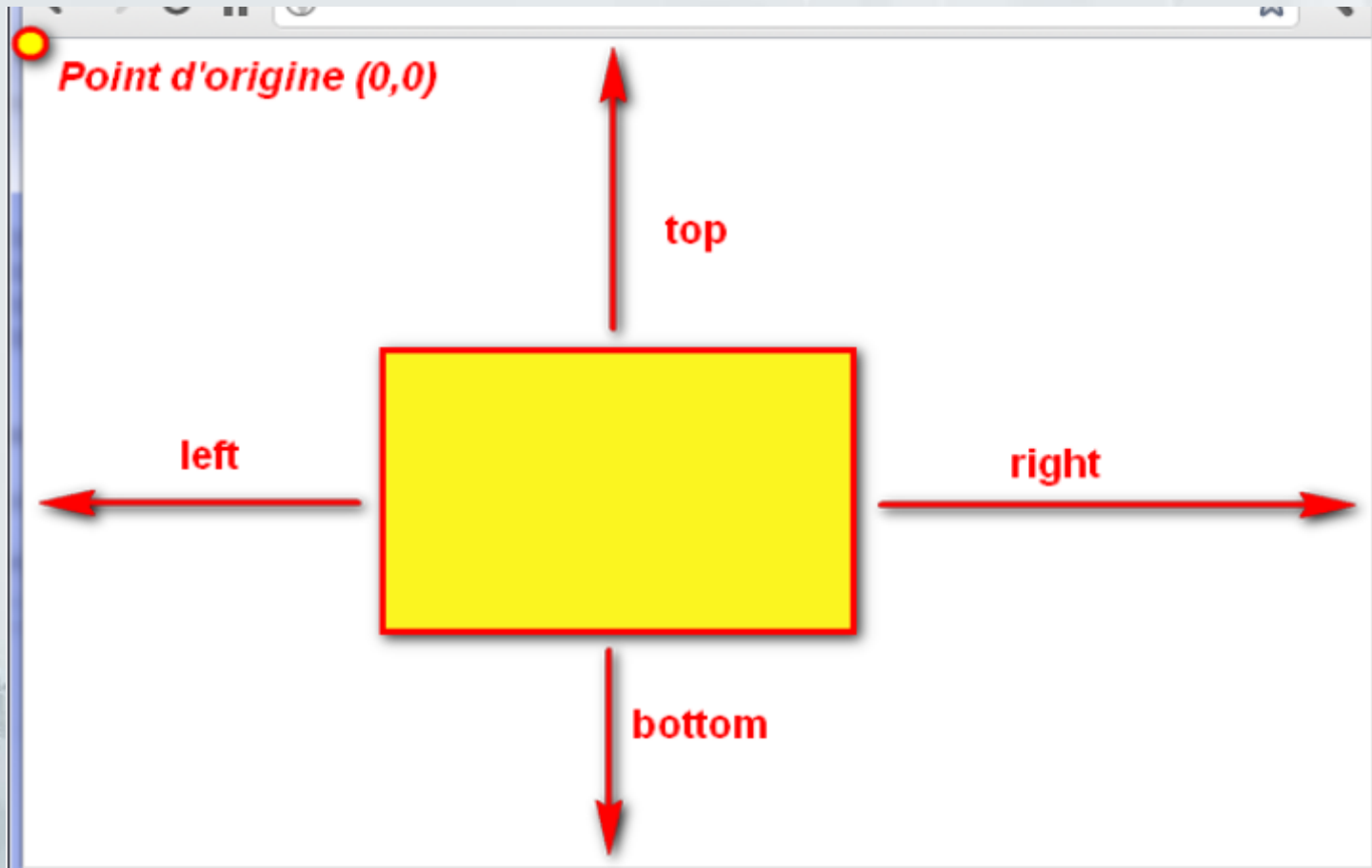




Mais cela ne suffit pas ! On a dit qu'on voulait un positionnement absolu, mais encore faut-il dire où l'on veut que le bloc soit positionné sur la page.

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.



Avec cela, vous devriez être capable 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 :

```
1 element
2 {
3     position: absolute;
4     right: 0px;
5     bottom: 0px;
6 }
```

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



# salut !

## Carnets de voyage

### À propos de l'auteur

éy C'est moi ! ! Je suis né en 23  
novembre 2005 .

### Je suis un grand voyageur

Bla bla bla bla (texte de l'article)

Copyright nom - Tous droits réservés

[Me contacter !](#)

[Accueil](#)  
[Blog](#)  
[photos](#)

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.

```
1  element
2  {
3      position: absolute;
4      right: 0px;
5      bottom: 0px;
6      z-index: 1;
7  }
8  element2
9  {
10     position: absolute;
11     right: 30px;
12     bottom: 30px;
13     z-index: 2;
14 }
```

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.

```
1 element
2 {
3     position: fixed;
4     right: 0px;
5     bottom: 0px;
6 }
```

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).

Bla bla bla bla (texte de l'article)

Bla bla bla bla (texte de l'article)

Bla bla bla bla (texte de l'article)

Bla bla bla bla (texte de l'article)

Bla bla bla bla (texte de l'article)

Bla bla bla bla (texte de l'article)

Bla bla bla bla (texte de l'article)

Bla bla bla bla (texte de l'article)

Bla bla bla bla (texte de l'article)

Bla bla bla bla (texte de l'article)

Bla bla bla bla (texte de l'article)

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



## 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 :

```
1 strong
2 {
3     background-color: red; /* Fond rouge */
4     color: yellow; /* Texte de couleur jaune */
5 }
```

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.

point d'origine (0, 0)

Pas de doute, **ce texte est important** si on veut comprendre corr

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) :

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

Pas de doute,  **ce texte est important** si on veut com

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

# Pratiquez !

## Partie 1:



**Le voyageur !**

ACCUEIL BLOG CV CONTACT



Retour sur mes vacances aux États-Unis...

[Voir l'article ►](#)





## Partie 2:



### JE SUIS UN GRAND VOYAGEUR

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam nec sagittis massa. Nulla facilisi. Cras id arcu lorem, et semper purus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis vel enim mi, in lobortis sem. Vestibulum luctus elit eu libero ultrices id fermentum sem sagittis. Nulla imperdiet mauris sed sapien dignissim id aliquam est aliquam. Maecenas non odio ipsum, a elementum nisi. Mauris non erat eu erat placerat convallis. Mauris in pretium urna. Cras laoreet molestie odio, consequat consequat velit commodo eu. Integer vitae lectus ac nunc posuere pellentesque non at eros. Suspendisse non lectus lorem.

Vivamus sed libero nec mauris pulvinar facilisis ut non sem. Quisque mollis ullamcorper diam vel faucibus. Vestibulum sollicitudin facilisis feugiat. Nulla euismod sodales hendrerit. Donec quis orci arcu. Vivamus fermentum magna a erat ullamcorper dignissim pretium nunc aliquam. Aenean pulvinar condimentum enim a dignissim. Vivamus sit amet lectus at ante adipiscing adipiscing eget vitae felis. In at fringilla est. Cras id velit ut magna rutrum commodo. Etiam ut scelerisque purus. Duis risus elit, venenatis vel rutrum in, imperdiet in quam. Sed vestibulum, libero ut bibendum consectetur, eros ipsum ultrices nisl, in rutrum diam augue non tortor. Fusce nec massa et risus dapibus aliquam vitae nec diam.

Phasellus ligula massa, congue ac vulputate non, dignissim at augue. Sed auctor fringilla quam quis porttitor. Praesent vitae dignissim magna. Pellentesque quis sem purus, vel elementum mi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Maecenas consectetur euismod urna. In hac habitasse platea dictumst. Quisque tincidunt porttitor vestibulum. Ut iaculis, lacus at molestie lacinia, ipsum mi adipiscing ligula, vel mollis sem risus eu lectus. Nunc elit quam, rutrum ut dignissim sit amet, egestas at sem.

### À PROPOS DE L'AUTEUR



Je me présente : je m'appelle Lego, je suis né un 23 novembre 1977.

J'aime la vie tranquille, et la mienne est agitée par une infinité de détails turbulents.



## Partie 3:

### MON DERNIER TWEET

Hii haaaaaan !

le 12 mai à 23h12

### MES PHOTOS



### MES AMIS

- Pupi le lapin
- Mr Baobab
- Kaiwaii
- Perceval.eu
- Belette
- Le concombre masqué
- Ptit prince
- Mr Fan



# Pratiquez !

## Chabout Marwa



Formatrice & développeur web full stack

### Mes expériences

- **De 2018-2016** : fondateur du Site du anaya , site de produits cosmétique en ligne . Animation le soir et le week-end en parallèle de mes études.
- **De 2019 à 2020** : Certification en International Software Testing Qualifications Board
- **De 2019 à 2020** : Certification en scrum master product owner
- **De 2021 à aujourd'hui** : transformation de Simple IT et du

### Mes compétences

- HTML5 et CSS3
- PHP5,Symfony,Angular ([certificat](#))
- Mysql/Sql
- Super Smash Bros (niveau expert)

### Ma formation

Alors là c'est simple... j'ai tout appris avec notre formatrice !!! !

