



# Cours HTML

## 3WAcademy

Cours d'intégration Web

---

### 3. La mise en page

#### 3.1. Le modèle de boîte

#### 3.2. Propriétés de base

##### 3.2.1. content

##### 3.2.2. padding

##### 3.2.3. border

##### 3.2.4. margin

#### 3.3. La propriété display

#### 3.4. Les propriétés avancées

##### 3.4.1. box-sizing

##### 3.4.2. overflow

#### 3.5. Le flux

#### 3.6. Le positionnement

##### 3.6.1 position: static

##### 3.6.2 position: relative

##### 3.6.3 position: absolute

##### 3.6.4 position: fixed

##### 3.6.5 position: sticky

#### 3.7. Le modèle flexbox

##### 3.7.1. Présentation

##### 3.7.2. flex-direction (conteneur)

##### 3.7.3. flex-wrap (conteneur)

##### 3.7.4. align-items (conteneur)

##### 3.7.4. justify-content (conteneur)

##### 3.7.5. propriétés appliquées aux items

## 3. La mise en page

### 3.1. Le modèle de boîte

En CSS le modèle de boîte est la base de la mise en page. Il décrit chaque élément comme une boîte rectangulaire qui possède une série de propriétés liée à la structure en pelure d'oignons de cette boîte. Elle est ainsi décomposée en quatre contours imbriqués: la marge (margin), la bordure (border), le remplissage (padding) et le contenu (content).



### 3.2. Propriétés de base

Chaque couche de la boîte peut être modifiée avec des propriétés CSS spécifiques:

#### 3.2.1. content

La taille du contenu dépend du texte qu'il contient (sa largeur dépend du parent et sa hauteur est déduite du nombre de ligne de texte) mais peut être modifiée avec les propriétés

*width* (largeur) et *height* (hauteur) mais aussi avec leurs déclinaisons minimale et maximale (*min-width*, *max-width*, *min height* et *max-height*).

Dans le CSS

```
.example {  
    width: 50%;  
    min-width: 200px;  
}
```

### 3.2.2. padding

Le remplissage permet de mettre de l'espace entre le contenu et la bordure pour améliorer la lisibilité (le texte qui colle à son encadrement ne sera pas très lisible). Il peut être modifié avec la propriété *padding* ou ses déclinaisons en fonction:

- du haut: *padding-top*,
- du bas: *padding-bottom*,
- du côté gauche: *padding-left*,
- du côté droit: *padding-right*.

Dans le CSS

```
.example {  
    padding: 1em;  
}
```

### 3.2.3. border

La bordure permet d'afficher une décoration autour de la boîte. Elle peut être modifiée avec la propriété *border* ou ses déclinaisons en fonction:

- du haut: *border-top*,
- du bas: *border-bottom*,
- du côté gauche: *border-left*,
- du côté droit: *border-right*.

La propriété *border* contient trois composantes:

- sa taille en px ou em
- son style: plein (solid), en tirets (dashed), en pointillés (dotted) ...
- sa couleur

Dans le CSS

```
.example {  
    border: 1px solid red;  
}
```

```
}
```

### 3.2.4. margin

La marge permet de mettre de l'espace entre la boîte et les autres boîtes adjacentes. Elle peut être modifiée avec la propriété *margin* ou ses déclinaisons en fonction:

- du haut: *margin-top*,
- du bas: *margin-bottom*,
- du côté gauche: *margin-left*,
- du côté droit: *margin-right*.



**Attention** les marges des boîtes successives se recouvrent, elles ne s'additionnent pas.

Dans le CSS

```
.example {  
    margin-bottom: 4em;  
}
```

## 3.3. La propriété display

Certainement la propriété la plus importante de CSS. Elle permet de modifier l'algorithme régissant la disposition de l'élément concerné par rapport aux éléments voisins.

Voici quelques valeurs possibles de la propriété *display*:

<i>block</i>	<p>L'élément prend toute la largeur disponible, sa largeur correspondant à l'espace disponible de son parent. Il y a un saut de ligne implicite avant l'élément et un autre après. Les propriétés liées au modèle de boîte s'appliquent sans restriction.</p> <p>C'est la valeur par défaut pour les éléments P, H1, H2, H3 ...</p>
<i>inline</i>	<p>L'élément se comporte comme un mot dans un texte. La largeur de l'élément est celle de son contenu. L'élément se place à la suite de ses voisins de type <i>inline</i>, sur la même ligne. Un espace est automatiquement généré entre plusieurs éléments de type <i>inline</i>. Les propriétés liées au modèle de boîte sont partiellement gérées.</p> <p>C'est la valeur par défaut pour les éléments A, STRONG, EM ...</p>

<i>inline-block</i>	L'élément se comporte comme un élément <i>inline</i> vis à vis de ses voisins mais se comporte comme un bloc en interne: les propriétés liées aux modèle de boîte s'applique sans restriction en interne.
<i>none</i>	L'élément n'est plus affiché.
<i>table</i>	L'élément se comporte comme une table.  C'est la valeur par défaut pour l'élément TABLE
<i>flex</i>	L'élément se comporte comme un bloc mais applique à ses éléments enfant une disposition suivant le modèle "flexbox".
<i>grid</i>	L'élément se comporte comme un bloc mais applique à ses éléments enfant une disposition suivant le modèle "grid".

Tout élément a une valeur `display` prédéfini, il n'y a pas de valeur par défaut globale.

## 3.4. Les propriétés avancées

### 3.4.1. box-sizing

Cette propriété permet de modifier les couches du modèle de boîte pris en compte dans le calcul de la taille (*width*). Par défaut la valeur est *content-box*, la largeur de la boîte est celle de son "content" et ne comprend pas la largeur du *padding* ou la largeur du *border*, il est possible déclarer la valeur *border-box* qui à l'inverse, inclut ces largeurs.

Dans le CSS

```
* {
  box-sizing: border-box;
}
```

### 3.4.2. overflow

L'*overflow* gère le dépassement du contenu d'une boîte par rapport à l'espace alloué de sa zone "content". Par exemple, si le contenu d'une boîte prend plus de place que l'espace qui est disponible, à cause d'une contrainte, alors, c'est le comportement par défaut, le contenu dépasse de sa propre boîte et peut se superposer au contenu des boîtes voisines. On peut contrôler ce comportement avec les différentes valeurs de la propriété *overflow*:

visible	Le contenu qui dépasse est visible, c'est la valeur par défaut.
hidden	Le contenu qui dépasse est caché.
scroll	Des barres de défilement apparaissent sur les côtés de la boîte qui permettent d'afficher le contenu qui dépasse.
auto	Une ou plusieurs barres de défilement apparaissent en fonction des besoins sur le côté de la boîte pour afficher le contenu qui dépasse.

Dans le CSS

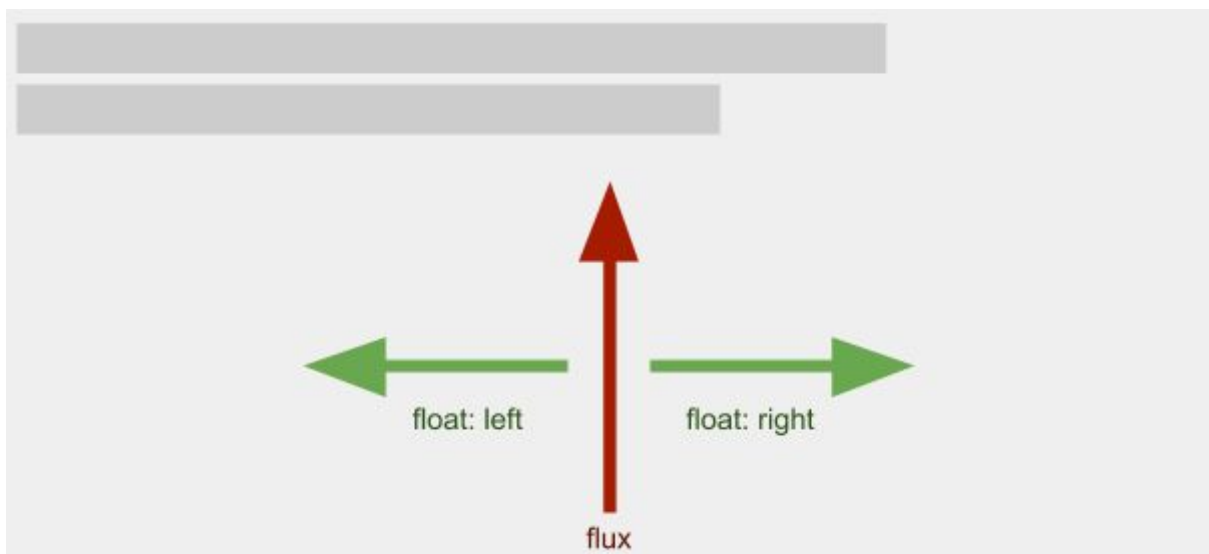
```
.example {
  max-height: 20em;
  overflow: auto;
}
```

### 3.5. Le flux

Le flux HTML peut se définir comme étant le comportement naturel d'affichage des éléments d'une page web. Autrement dit, les éléments se placent les uns à la suite des autres depuis le haut de la page. Il en découle que la position de chaque élément dépend de celui d'avant.

Le flux agit comme une gravité inversée. On peut le schématiser par une flèche du bas vers le haut.

On peut faire sortir un élément du flux par la gauche ou par la droite à l'aide de la propriété `float`, par exemple pour faire en sorte que le texte entoure une image.



Dans le CSS

```
img.left {  
    float: left;  
    margin-right: 1em;  
}
```

Dans le document HTML

```
<p>  
      
    Lorem ipsum ...  
</p>
```

Un élément flottant n'est pas pris en compte dans le calcul de la hauteur de son parent, il faut parfois utiliser un élément additionnel avec la propriété `clear` ou un artifice tel que "clearfix" pour le prendre en compte à posteriori.

On a longtemps utilisé la propriété *float* pour faire de la mise en page tel que juxtaposer des éléments les uns à côté des autres, il est maintenant préférable d'utiliser des méthodes modernes telles que "flex" ou "grid".

## 3.6. Le positionnement

Il est possible de contourner le flux d'une autre façon, en appliquant la propriété *position* qui va permettre de donner à l'élément concerné une position en fonction de coordonnées en utilisant les propriétés *top*, *bottom*, *left* et *right* pour déclarer ces coordonnées. La valeur de ces coordonnées et l'origine dont elles dépendent sont fonction de la valeur choisie pour la propriété *position*.

### 3.6.1 position: static

C'est la valeur par défaut, la position dépend du flux normal du document. Aucune coordonnée n'est prise en compte.

### 3.6.2 position: relative

Dans ce cas, on peut déplacer l'élément par rapport à sa position initiale dans le flux.

Dans le CSS

```
.example {  
    position: relative;  
    top: 10px;  
    left: 20px;  
}
```

ici l'élément sera positionné plus bas de 0px et plus à droite de 20px par rapport à sa position initiale dans le flux.

### 3.6.3 position: absolute

Dans ce cas, on peut déplacer l'élément par rapport à la position du premier élément parent qui a la propriété position avec la valeur relative ou absolute. L'élément BODY est configuré par défaut comme étant position relative ce qui en fait une origine par défaut des éléments ainsi positionné.

Dans le CSS

```
.example {  
    position: absolute;  
    top: 10px;  
    left: 20px;  
}
```

ici, en l'absence d'autres éléments ayant la propriété position modifiée, l'élément *.example* sera positionné à, respectivement, 10px du haut et 20px du côté gauche du document.

### 3.6.4 position: fixed

Dans ce cas le positionnement se fait par rapport aux bord du viewport, c'est à dire du cadre d'affichage du navigateur internet. c'est le cas de parties du document qui collent au bord de la fenêtre du navigateur même si l'on scrolle le document.

### 3.6.5 position: sticky

Dans ce cas le positionnement se fait par rapport aux bord du viewport, c'est à dire du cadre d'affichage du navigateur internet mais seulement si l'élément est en dehors de celui ci. En général ce comportement s'observe lorsque l'on scrolle le document et que l'on observe des parties qui collent aux bord alors qu'ils ne devraient plus être visibles.



## 3.7. Le modèle flexbox

### 3.7.1. Présentation

Le modèle flexbox permet de faire une mise en page très complexe d'une série d'éléments sans détourner d'autres propriétés CSS non conçue initialement pour cet usage (comme l'utilisation de `display table`, de `display inline-block` ou de `float`) et sans les inconvénients respectives de ces autres méthodes.

On nomme le parent (qui contient la série d'éléments à disposer) le "container" (conteneur en français) et les enfants de ce conteneur les "items". Les items sont représentés en rouge dans les schémas suivants. Certaines propriétés du modèle flexbox s'appliquent au conteneur, d'autres propriétés s'appliquent aux items.

Une simple instruction suffit pour transformer une liste verticale en liste horizontale. Cette propriété (*`display: flex`*) est donnée au conteneur.

Dans le CSS

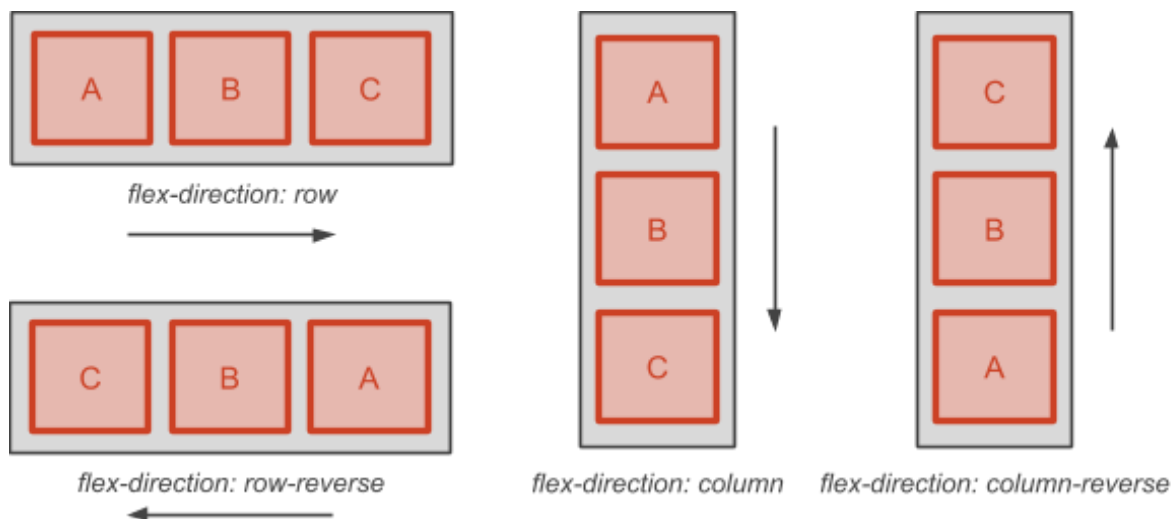
```
.example {  
    display: flex;  
}
```

Dans le document HTML

```
<ul class="example">  
  <li>Lorem ipsum ...</li>  
  <li>...</li>  
</ul>
```

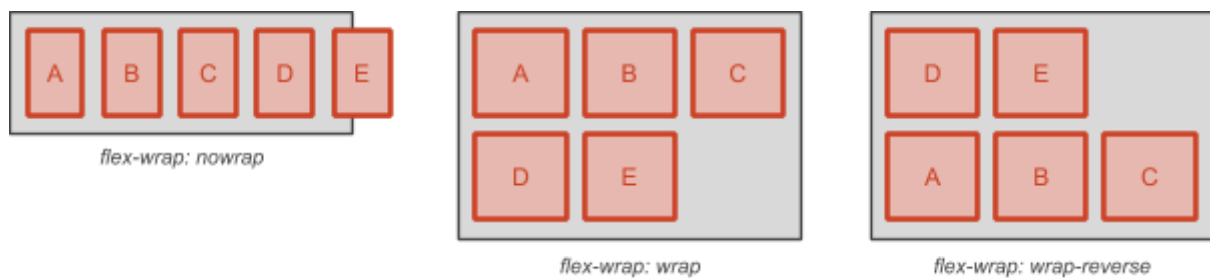
### 3.7.2. flex-direction (conteneur)

Il est possible de changer l'orientation principale avec la propriété `flex-direction`:



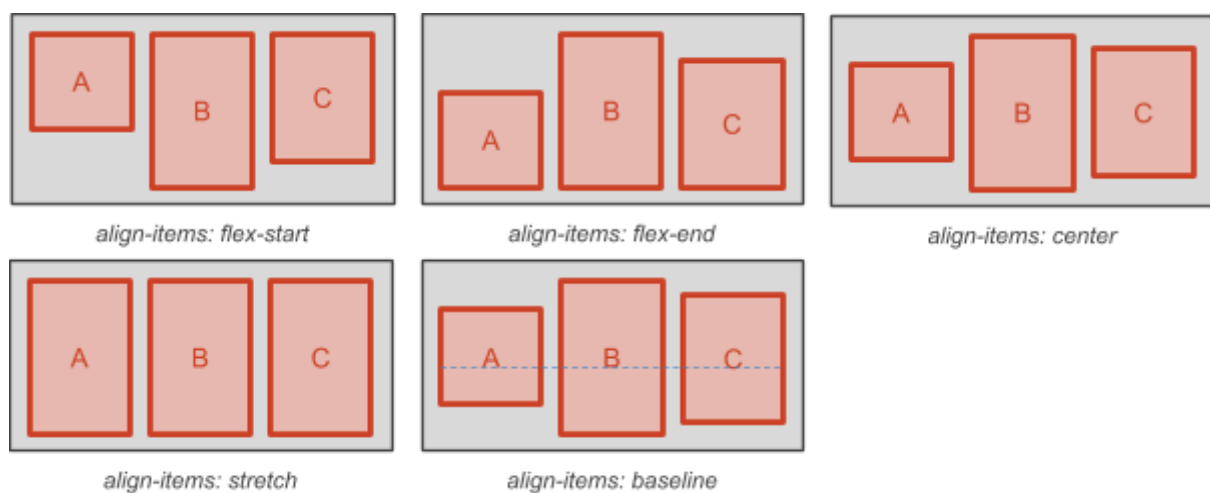
### 3.7.3. flex-wrap (conteneur)

Il est possible de disposer les items sur plusieurs lignes avec la propriété flex-wrap :



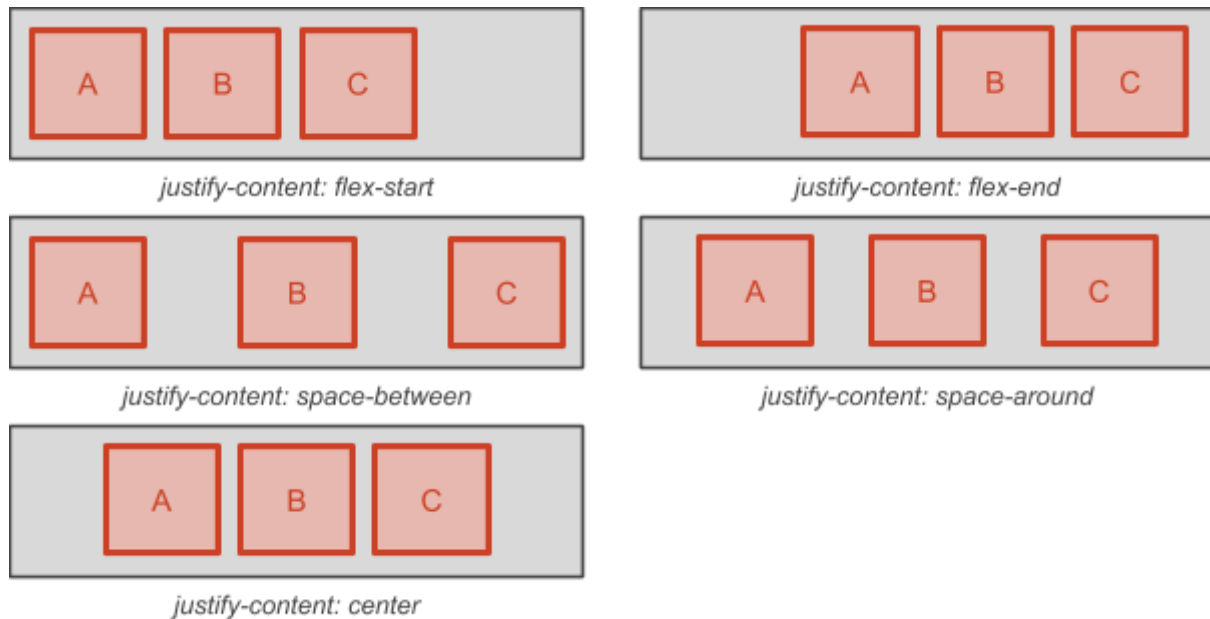
### 3.7.4. align-items (conteneur)

Il est possible d'ajuster l'alignement vertical des items avec la propriété align-items:



### 3.7.4. justify-content (conteneur)

Il est possible de répartir les items horizontalement avec la propriété justify-content:



Il est aussi possible de répartir les items verticalement avec la propriété align-content et les mêmes valeurs.

### 3.7.5. propriétés appliquées aux items

Un certain nombre de propriétés du modèle flexbox s'applique directement aux items:

*align-self* permet d'aligner un item avec les mêmes valeur que la propriété du conteneur *align-items*.

*flex-grow* permet de spécifier le poids (grossissant) d'un item par rapport aux autres dans la répartition des tailles des items.

*flex-shrink* permet de spécifier le poids (amincissant) d'un item par rapport aux autres dans la répartition des tailles des items.

*order* permet de modifier l'ordre de chacun des items.