

Français ▼

Les concepts de base pour flexbox

Le module des boîtes flexibles, aussi appelé « *flexbox* », a été conçu comme un modèle de disposition unidimensionnel et comme une méthode permettant de distribuer l'espace entre des objets d'une interface ainsi que de les aligner. Dans cet article, nous verrons les fonctionnalités principales des *flexbox* que nous approfondirons ensuite dans d'autres articles.

Lorsqu'on décrit les boîtes flexibles comme une méthode de disposition unidimensionnelle, on indique en fait que les *flexbox* gèrent une seule dimension à la fois : une ligne ou une colonne. Ce modèle est à comparer au modèle bidimensionnel de la disposition en grille (CSS Grid) qui contrôle à la fois les colonnes et les lignes.

Les deux axes des boîtes flexibles

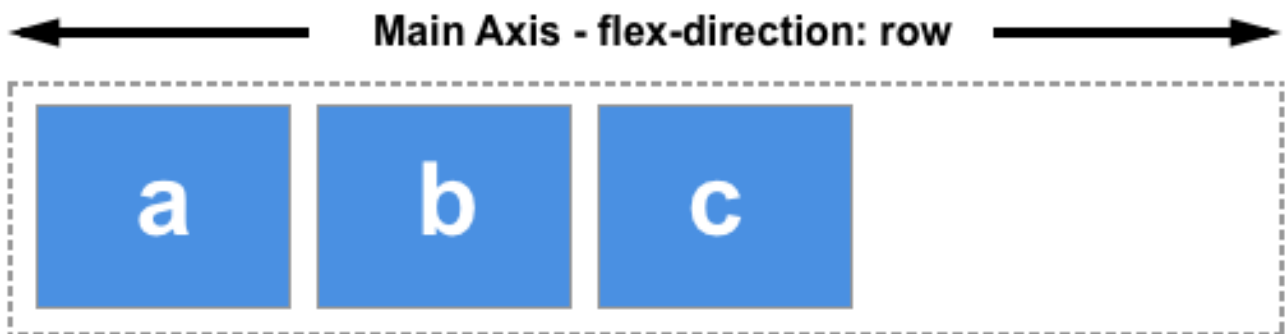
Lorsqu'on travaille avec les boîtes flexibles, deux axes interviennent : l'axe principal (*main axis* en anglais) et l'axe secondaire (*cross axis* en anglais). L'axe principal est défini par la propriété `flex-direction` et l'axe secondaire est l'axe qui lui est perpendiculaire. Tout ce que nous manipulons avec les boîtes flexibles fera référence à ces axes.

L'axe principal

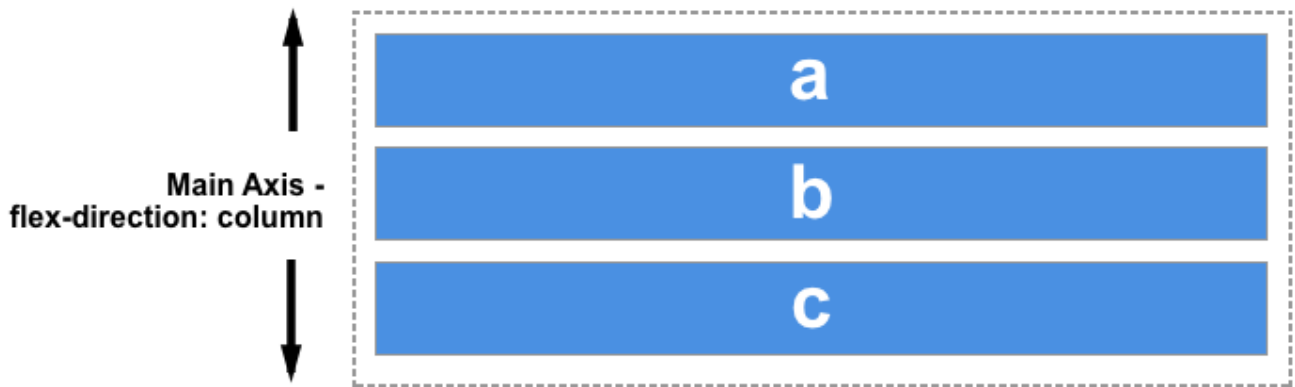
L'axe principal est défini par la propriété `flex-direction` qui peut prendre quatre valeurs :

- `row`
- `row-reverse`
- `column`
- `column-reverse`

Si on choisit la valeur `row` ou `row-reverse`, l'axe principal sera aligné avec la direction « en ligne » (*inline direction*) (c'est la direction logique qui suit le sens d'écriture du document).



Si on choisit la valeur `column` ou `column-reverse`, l'axe principal suivra la direction de bloc (*block direction*) et progressera le long de l'axe perpendiculaire au sens d'écriture.

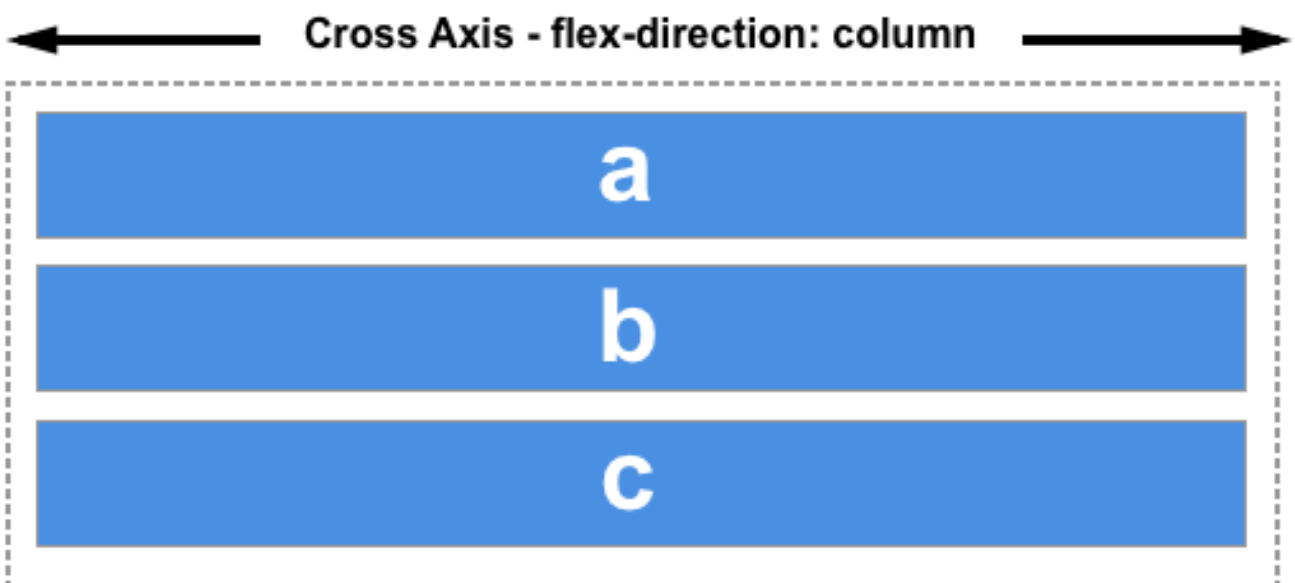


L'axe secondaire (*cross axis*)

L'axe secondaire est perpendiculaire à l'axe principal. Ainsi, si flex-direction vaut row ou row-reverse, l'axe secondaire suivra l'axe des colonnes.



Si l'axe principale est column ou column-reverse, l'axe secondaire suivra celui des lignes (horizontales).



Comprendre les liens entre les différents axes est crucial lorsqu'on commence à aligner/justifier des éléments flexibles sur un axe ou l'autre grâce aux fonctionnalités et propriétés des boîtes flexibles.

Les lignes de début et de fin

Une autre notion fondamentale est l'absence d'hypothèse sur le mode d'écriture du document. Pour les modèles de disposition précédents, CSS était fortement orienté vers les modes d'écritures de droite à gauche et de gauche à droite. Les modèles de disposition modernes permettent de gérer naturellement les différents modes d'écriture et ne reposent plus sur l'idée qu'une ligne de texte commencera en haut à gauche d'un document puis progressera vers la droite et que chaque nouvelle ligne apparaîtra sous la précédente.

Nous verrons plus tard les détails des relations entre les spécifications des boîtes flexibles et celles des modes d'écriture. Toutefois, décrivons ici pourquoi on ne parlera plus de gauche ou de droite et de bas ou de haut lorsque nous évoquerons la direction dans laquelle s'organisent les éléments flexibles.

Si `flex-direction` vaut `row` et que nous travaillons sur un document écrit en français, la ligne de début de l'axe principal sera située à gauche et la ligne de fin sera située à droite.



Si on travaille sur un document écrit dans une langue arabe, la ligne de début de l'axe principal sera à droite et la ligne de fin à gauche.



Dans les deux cas, la ligne de début de l'axe secondaire est située en haut et la ligne de fin de cet axe est située en bas car ces deux langues sont écrites horizontalement.

Nous verrons qu'au fur et à mesure, il devient naturel de parler de début et de fin plutôt que de gauche et de droite. De plus, ce niveau d'abstraction sera utile pour comprendre d'autres méthodes de disposition comme les grilles CSS car il y est également utilisé.

Le conteneur flexible


La zone d'un document sujette au modèle de disposition *flexbox* est appelée un **conteneur flexible**. Pour créer un conteneur flexible, il faut que la valeur de la propriété `display` de cet élément soit `flex` ou

`inline-flex`. Dès que c'est le cas, les éléments « enfants » directs deviennent des **éléments flexibles** (***flex items***). Comme pour les autres propriétés CSS, certaines valeurs initiales sont définies, aussi, lorsqu'on crée un conteneur flexible, tous les éléments flexibles se comporteront de la façon suivante :

- Les éléments s'afficheront en lignes horizontales (la valeur par défaut de la propriété `flex-direction` est `row`).
- Les éléments seront placés à partir de la ligne de début de l'axe principal.
- Les éléments ne s'étireront pas le long de l'axe principal mais pourront se rétrécir si nécessaire.
- Les éléments seront étirés le long de l'axe secondaire afin d'occuper l'espace sur cet axe.
- La propriété `flex-basis` vaut `auto`.
- La propriété `flex-wrap` vaut `nowrap`.

Autrement dit, tous les éléments formeront une ligne en utilisant la taille de leur contenu. S'il y a plus d'éléments que le conteneur peut en contenir, ils ne formeront pas une nouvelle ligne mais dépasseront du conteneur. Si certains éléments sont plus grands (selon l'axe secondaire) que d'autres, tous les éléments s'étireront sur l'axe secondaire afin d'avoir la plus grande taille.

Vous pouvez étudier l'exemple qui suit pour voir le résultat obtenu. N'hésitez pas à éditer les éléments ou à en ajouter d'autres pour tester ce comportement initial.



The diagram shows a horizontal flex container. Inside, there are three items: a box labeled 'One', a box labeled 'Two', and a box labeled 'Three' followed by the text 'has extra text' on two lines. The items are arranged horizontally from left to right.

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

```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three  
    <br>has  
    <br>extra  
    <br>text  
  </div>  
</div>
```

Reset

Modifier flex-direction

En ajoutant la propriété `flex-direction` au conteneur flexible, on peut modifier la direction dans laquelle les éléments flexibles seront affichés. En utilisant `flex-direction: row-reverse`, les éléments seront affichés le long d'une ligne horizontale mais les lignes de début et de fin seront inversées.

Si on utilise `column` comme valeur de `flex-direction`, l'axe principal est modifié et les éléments sont affichés sur une colonne. Si on utilise `column-reverse`, les lignes de début et de fin seront également inversées.

Dans l'exemple suivant, on utilise `flex-direction` avec la valeur `row-reverse`. Vous pouvez utiliser d'autres valeurs — `row`, `column` et `column-reverse` — afin de voir le résultat produit.



```
.box {  
  display: flex;  
  flex-direction: row-reverse;  
}
```

```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
</div>
```

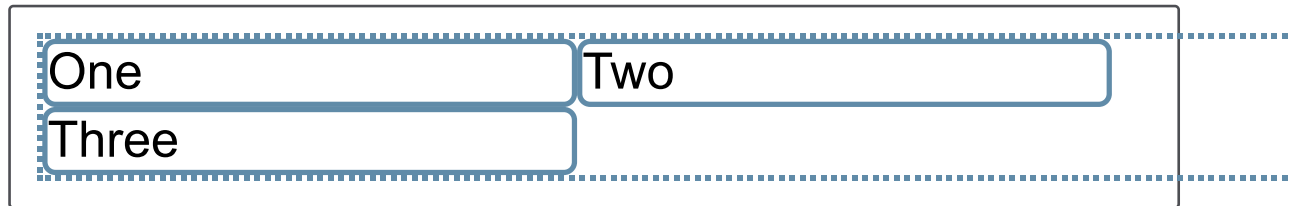
Reset

Créer un conteneur flexible sur plusieurs lignes avec `flex-wrap`

Bien que le modèle des boîtes flexibles soit organisé sur une dimension, il est possible d'organiser les éléments flexibles afin que ceux-ci s'étendent sur plusieurs lignes ou colonnes (plutôt que de dépasser). Lorsque c'est le cas, chaque nouvelle ligne ou colonne agit comme un nouveau conteneur flexible. La distribution de l'espace sur cette ligne/colonne ne tiendra pas compte des autres lignes/colonnes.

Pour obtenir ce « passage à la ligne », on ajoute la propriété `flex-wrap` avec la valeur `wrap`. Désormais, si les éléments sont trop grands pour tenir sur une seule ligne, ils passeront sur une autre ligne.

L'exemple suivant illustre le résultat obtenu lorsque la somme des tailles des éléments dépasse celle du conteneur. Avec `flex-wrap` qui vaut `wrap`, les éléments passent à la ligne. Si on modifie la valeur avec `nowrap` (qui correspond à la valeur initiale), les éléments seront rétrécis pour tenir sur une ligne (car les valeurs initiales des boîtes flexibles permettent aux éléments d'être ainsi redimensionnés). Si on utilise `nowrap` et que les éléments ne peuvent pas être redimensionnés (ou pas suffisamment), cela causera un dépassement.



```
.box {  
  display: flex;  
  flex-wrap: wrap;  
}
```

```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
</div>
```

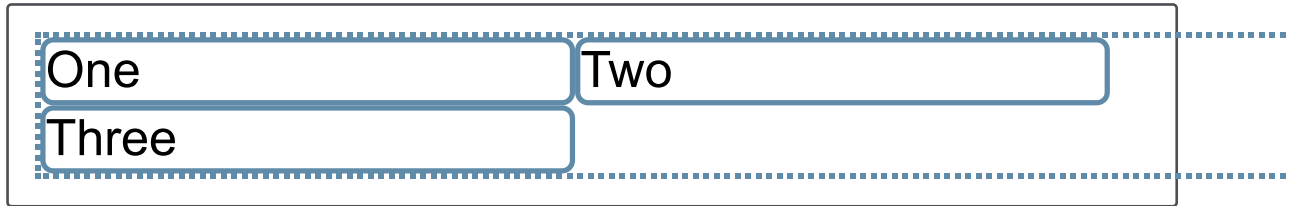
Reset

Pour approfondir ces notions, vous pouvez consulter l'article [Maîtriser le passage à la ligne des éléments flexibles](#).

La propriété raccourcie flex-flow

Il est possible de synthétiser les propriétés flex-direction et flex-wrap avec la propriété raccourcie flex-flow. La première valeur de cette propriété sera utilisée pour flex-direction et la seconde pour flex-wrap.

Dans l'exemple qui suit, vous pouvez changer les valeurs de `flex-direction` en utilisant `row`, `row-reverse`, `column` ou `column-reverse` pour la première et `wrap` ou `nowrap` pour la seconde.



```
.box {  
  display: flex;  
  flex-flow: row wrap;  
}
```

```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
</div>
```

Reset

Les propriétés appliquées aux éléments flexibles

Pour mieux contrôler les éléments flexibles, on peut les cibler directement avec trois propriétés :

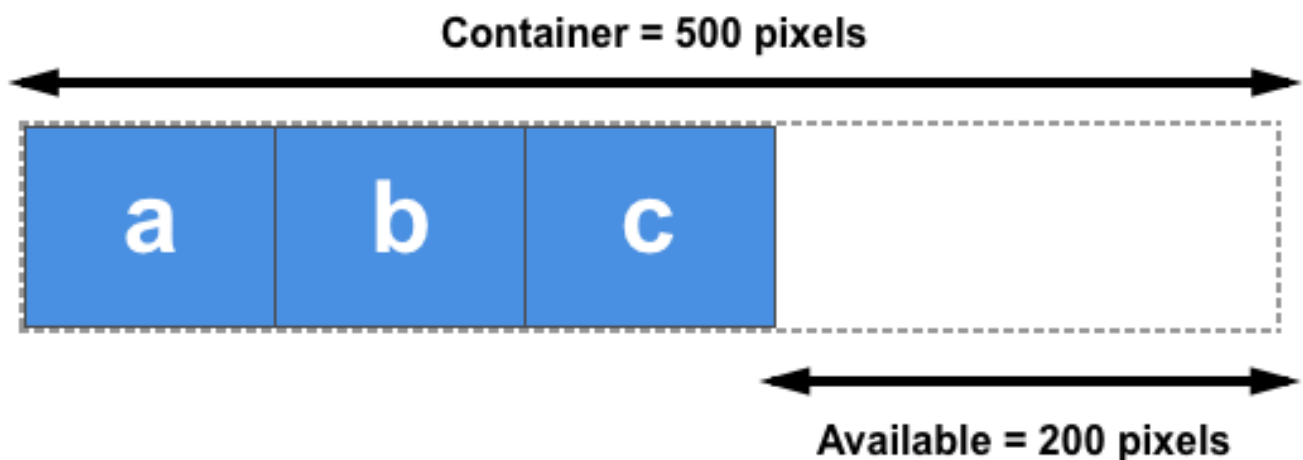
- `flex-grow`

- `flex-shrink`
- `flex-basis`

Nous verrons ici un rapide aperçu de ces propriétés que nous approfondirons dans l'article [Contrôler les proportions des éléments flexibles le long de l'axe principal](#).

Avant de revenir à ces propriétés, il nous faut définir le concept d'**espace disponible**. Lorsque nous modifierons l'une de ces propriétés, cela modifiera la façon dont l'espace disponible est distribué parmi les éléments. Ce concept est également important lorsqu'on aligne les éléments.

Prenons un conteneur de 500 pixels de large et qui contient trois éléments de 100 pixels de large. Il faut donc 300 pixels pour disposer ces éléments et il reste ainsi 200 pixels d'espace disponible. Si on ne modifie pas les valeurs initiales, l'espace disponible sera placé après le dernier élément.



Si on préfère que les éléments s'étirent pour occuper l'espace restant, il nous faut une méthode pour distribuer cet espace parmi les éléments. C'est le rôle des propriétés `flex-` qui s'appliquent aux éléments.

La propriété `flex-basis`

La propriété `flex-basis` définit la taille de l'élément en termes d'espace occupé. La valeur initiale de cette propriété est `auto` — dans ce cas, le navigateur analyse si les éléments ont une taille. Dans l'exemple précédent, les éléments mesurent 100 pixels de large et c'est donc cette mesure qui est utilisée pour `flex-basis`.

Si les éléments n'ont pas de taille définie, c'est la taille du contenu qui est utilisée comme base. C'est pour ça que nous avons simplement déclaré `display: flex` sur l'élément parent afin de créer des éléments flexibles (qui prennent alors tout l'espace nécessaire à leur contenu).

La propriété `flex-grow`

La propriété `flex-grow` est un entier positif qui, lorsqu'elle est définie, permet aux éléments flexibles de s'étendre à partir de la mesure de `flex-basis`. Ainsi, l'élément sera étiré et occupera l'espace disponible sur cet axe ou une part de cet espace si les autres éléments peuvent s'étendre également.

Si on utilise `flex-grow: 1` pour les différents éléments de l'exemple précédent, l'espace disponible sera alors partagé de façon égale entre les éléments qui seront alors étirés pour occuper l'ensemble du conteneur le long de l'axe principal.

La propriété `flex-grow` permet de répartir l'espace disponible en « parts ». Si, pour le premier élément, on indique `flex-grow` avec une valeur de 2 et, pour les autres éléments, `flex-grow` avec une valeur

de 1, deux « parts » de l'espace disponible seront données au premier élément (il recevra donc 100 pixels parmi les 200 pixels restants) et une part sera fournie à chacun des autres éléments (soit 50 pixels chacun parmi les 200 pixels restants).

La propriété `flex-shrink`

La propriété `flex-grow` permet de gérer la façon dont l'espace est ajouté sur l'axe principal. La propriété `flex-shrink` permet quant à elle de contrôler la façon dont l'espace est réduit. S'il n'y a pas suffisamment d'espace dans le conteneur pour disposer les éléments et que `flex-shrink` est un entier positif, l'élément peut alors devenir plus petit que la taille définie par `flex-basis`. De façon analogue à `flex-grow`, il est possible d'affecter différents coefficients aux différents éléments afin que ceux-ci rétrécissent plus fortement que d'autres. Plus la valeur de `flex-shrink` sera élevée, plus l'élément ciblé rétrécira (si les éléments voisins ont une valeur de `flex-shrink` plus faibles).

La taille minimale de l'élément sera prise en compte lors du rétrécissement. Cela signifie que `flex-shrink` peut être moins cohérent que `flex-grow` selon les cas aux limites. Nous verrons plus en détails comment cet algorithme fonctionne dans l'article [Contrôler les proportions des éléments le long de l'axe principal](#).

Note : Les valeurs de `flex-grow` et `flex-shrink` sont des proportions. Autrement dit, si tous les éléments ont `flex: 1 1 200px` et qu'on souhaite qu'un d'eux grandissent deux fois plus, on utiliserait `flex: 2 1 200px` pour cet élément. Mais avoir `flex: 10 1 200px` d'une part et `flex: 20 1 200px` d'autre part fonctionnerait exactement de la même façon.

La propriété raccourcie `flex` et les valeurs synthétiques

On voit rarement `flex-grow`, `flex-shrink` et `flex-basis` utilisées individuellement mais plutôt combinées avec la propriété raccourcie `flex`. La propriété raccourcie `flex` permet de définir les valeurs de cette propriété dans cet ordre : `flex-grow`, `flex-shrink`, `flex-basis`.

L'exemple suit vous permet de tester différentes valeurs pour `flex`. La première valeur est `flex-grow` et un coefficient positif permettra à l'élément de grandir, la deuxième valeur est `flex-shrink` et un coefficient positif permettra de rétrécir l'élément s'il dépasse du conteneur sur l'axe principal. Enfin, la troisième valeur sert à `flex-basis` qui indique la taille de base à partir de laquelle l'élément sera étendu ou rétréci.

One

Two

Three

```
.box {
  display: flex;
}

.one {
  flex: 1 1 auto;
}

.two {
  flex: 1 1 auto;
}

.three {
  flex: 1 1 auto;
}

<div class="box">
  <div class="one">One</div>
  <div class="two">Two</div>
```

Cette propriété permet également d'utiliser des valeurs synthétiques qui couvrent la majorité des scénarios. Vous verrez souvent ces valeurs utilisées dans les tutoriels et, dans de nombreux cas, celles-ci suffiront :

- `flex: initial`
- `flex: auto`
- `flex: none`
- `flex: <nombre-positif>`

Avec `flex: initial`, les éléments récupèrent les valeurs initiales pour les différentes propriétés du modèle de boîte flexible. Cette valeur permettra d'obtenir le même comportement que `flex: 0 1 auto`. Ici, `flex-grow` vaut 0 et les éléments ne s'agrandiront pas au-delà de la taille `flex-basis`. `flex-shrink` vaut 1 et les éléments pourront

rétrécir si besoin plutôt que de dépasser du conteneur. `flex-basis` vaut `auto` et les éléments utiliseront donc la taille qui leur a été définie sur l'axe principale ou la taille déterminée à partir du contenu.

Avec `flex: auto`, on obtient le même comportement que `flex: 1 1 auto`, la seule différence avec `flex: initial` est que les éléments peuvent s'étirer si besoin.

Avec `flex: none`, les éléments ne seront pas flexibles. Cette valeur est synonyme de `flex: 0 0 auto`. Les éléments ne peuvent ni s'agrandir, ni se rétrécir mais seront disposés avec `flex-basis: auto`.

On voit aussi souvent des valeurs comme `flex: 1` ou `flex: 2`, etc. Cela correspond à `flex: 1 1 0`. Les éléments peuvent s'agrandir ou bien rétrécir à partir d'une taille de base égale à 0.

Vous pouvez utiliser ces valeurs synthétiques dans l'exemple suivant :

One

Two

Three

```
.box {  
  display: flex;  
}  
  
.one {  
  flex: 1;  
}  
  
.two {  
  flex: 1;  
}  
  
.three {  
  flex: 1;  
}  
  
<div class="box">  
  <div class="one">One</div>  
  <div class="two">Two</div>  
  <div class="three">Three</div>  
</div>
```

Reset

Alignement, justification et distribution de l'espace disponible entre les éléments

Une fonctionnalité majeure des boîtes flexibles est de permettre l'alignement et la justification des éléments le long des axes principal et secondaire tout en distribuant l'espace entre les éléments flexibles.

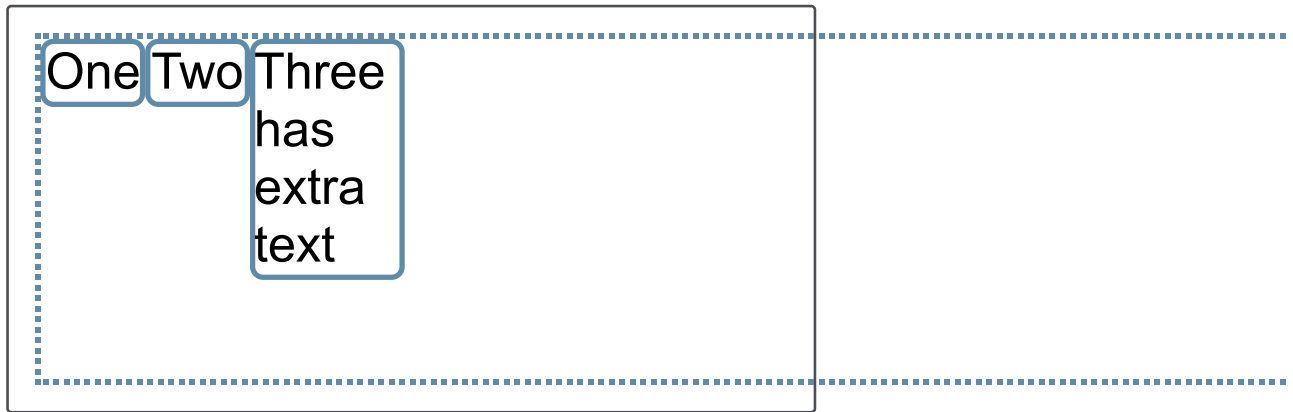
align-items

La propriété `align-items` permet d'aligner les éléments le long de l'axe secondaire.

La valeur initiale de cette propriété est `stretch`, ce qui explique pourquoi, par défaut, les éléments flexibles sont étirés sur l'axe perpendiculaire afin d'avoir la même taille que l'élément le plus grand dans cet axe (qui définit la taille du conteneur sur cet axe).

On peut également utiliser la valeur `flex-start` afin que les éléments soient alignés sur la ligne de début de l'axe secondaire, la valeur `flex-end` afin que les éléments soient alignés sur la ligne de fin de l'axe secondaire ou bien `center` pour les aligner au centre. Vous pouvez utiliser les valeurs suivantes dans l'exemple (on a donné un hauteur fixe au conteneur afin de pouvoir observer la façon dont les éléments se déplacent à l'intérieur) :

- `stretch`
- `flex-start`
- `flex-end`
- `center`



```
.box {  
  display: flex;  
  align-items: flex-  
start;  
}
```

```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three  
    <br>has  
    <br>extra  
    <br>text  
  </div>  
</div>
```

Reset


justify-content

La propriété `justify-content` est utilisée afin d'aligner les éléments le long de l'axe principal dans la direction définie par `flex-direction`. La valeur initiale est `flex-start` qui place les éléments à partir de la ligne de début du conteneur sur l'axe principal. La valeur `flex-end` permet de les placer vers la fin et la valeur `center` permet de les centrer le long de l'axe principal.

On peut également utiliser la valeur `space-between` afin de répartir l'espace disponible de façon égale entre chaque élément. Si on souhaite que l'espace soit également réparti autour des éléments, y compris au début et à la fin, on pourra utiliser la valeur `space-around` (il y aura alors un demi espace à la fin et au début). Si on souhaite que l'espace soit également réparti et qu'il y ait un espace entier au début et à la fin, on utilisera la valeur `space-evenly`.

Vous pouvez essayer les valeurs suivantes dans l'exemple suivant :

- `flex-start`
- `flex-end`
- `center`
- `space-around`
- `space-between`
- `space-evenly`



OneTwoThree

```
.box {  
  display: flex;  
  justify-content: flex-start;  
}
```

```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
</div>
```

Reset

Dans l'article [Aligner des éléments dans un conteneur flexible](#), nous verrons plus en détails comment ces propriétés fonctionnent. Ces premiers exemples permettent toutefois de comprendre comment utiliser ces propriétés sur une majorité de cas.

Prochaines étapes

Avec cet article, vous devriez comprendre les différentes fonctionnalités et concepts relatifs aux *flexbox*. Dans le prochain article, nous verrons [comment cette spécification s'articule avec les autres modules CSS](#).

Dernière modification : 6 avr. 2019, par des contributeurs MDN

Sujets associés

CSS

Référence CSS

CSS Flexible Box Layout

▼ Guides

Aligner des éléments dans un conteneur flexible

Rétrocompatibilité de flexbox

Les concepts de base pour flexbox

Contrôler les proportions des boîtes flexibles le long de l'axe principal

Mises en page avancées avec les boîtes flexibles

Maîtriser le passage à la ligne des éléments flexibles

Ordonner les éléments flexibles

Les liens entre flexbox et les autres méthodes de disposition

Cas d'utilisation classiques de flexbox

▼ Propriétés

`flex`

`flex-basis`

`flex-direction`

`flex-flow`

`flex-grow`

`flex-shrink`

flex-wrap

order



Recevez le meilleur du développement web

Recevez le meilleur de MDN, directement dans votre boîte de réception.

Cette lettre d'information est uniquement disponible en anglais pour l'instant.

Je m'inscris