

Flexbox

CSS Avancé

Table des matières

- Introduction
- Les deux axes des conteneurs flexibles
- Conteneur flexible sur plusieurs lignes
- Aligner les éléments
- Dimension dynamique des éléments
- Modifier l'ordre d'affichage

Introduction

Module révolutionnaire

Fini les bidouilles avec table, float, clear, position ...


Il permet :

- De gérer les alignements des boîtes,
- De centrer horizontalement et verticalement les boîtes,
- D'avoir des mises en page fluides,
- Des hauteurs de boîte similaires,
- Des fonds colorés uniformes,
- De concevoir des mises en page responsive.

Le support des Flexbox (caniuse.com)

CSS Flexible Box Layout Module - CR

Method of positioning elements in horizontal or vertical stacks.
 Support includes all properties prefixed with `flex`, as well as `display: flex`, `display: inline-flex`, `align-content`, `align-items`, `align-self`, `justify-content` and `order`.

Usage % of all users 

Global 93.2% + 3.47% = 96.68%

unprefixed 92.92% + 2.75% = 95.67%

Current aligned Usage relative Date relative Show all

IE	Edge	Firefox	Chrome	Safari	IOS Safari	Opera Mini	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			66		10.3				4
		60	67		11.2				6.2
11	17	61	68	11.1	11.4	all	67	11.8	7.2
	18	62	69	12	12				
		63	70	TP					
			71						

Le conteneur flexible

Les flexbox se basent sur l'utilisation d'un conteneur flexible (une div généralement) dans lequel les éléments enfants deviendront des « éléments flexibles ».

Pour créer le conteneur flexible, il faut modifier la valeur de la propriété CSS « display » pour que celle-ci soit « flex » ou « inline-flex ».

Exemple

Code HTML & CSS

```
<div id="conteneur">
  <div class="element" id="elem1">Elem 1</div>
  <div class="element" id="elem2">Elem 2</div>
  <div class="element" id="elem3">Elem 3</div>
</div>
```

```
#conteneur {
  display: flex;
  background-color: lightcyan;
  border: dashed 1px blue;
}

.element {
  background-color: greenyellow;
  border: solid 1px black;
  border-radius: 5px;
  margin: 1px;
}
```

Résultat



Les deux axes des conteneurs flexibles

L'axe principal (main axis)

L'axe principal est utilisé pour placer les éléments enfants dans le conteneur flexible. Celui-ci est défini par la propriété « flex-direction » qui peut prendre 4 valeurs :

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

Par défaut, la valeur de la propriété vaut « row ».

L'axe principal (main axis)

- row / row-reverse

Alignement avec la direction « inline »



- column / column-reverse

Alignement avec la direction « block »



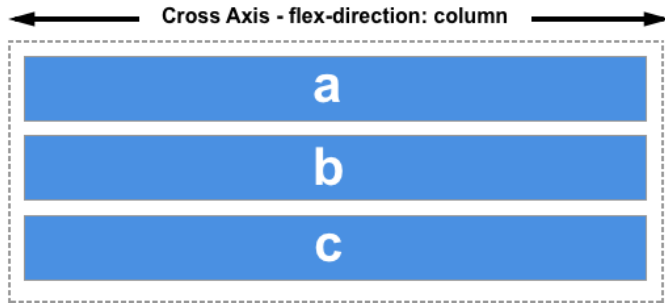
L'axe secondaire (cross axis)

L'axe secondaire est perpendiculaire à l'axe principal.

Si « flex-direction » vaut row où row-reverse, il s'aligne avec l'axe des colonnes.



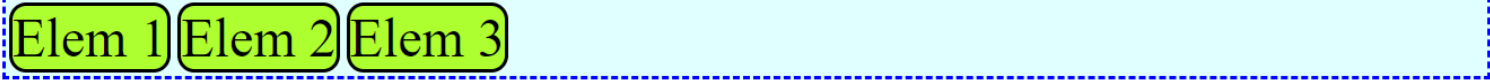
Si « flex-direction » vaut column où column-reverse, il s'aligne avec l'axe des lignes.



Exemple « flex-direction » (1/2)

- Row

```
#conteneur {  
  flex-direction: row;  
}
```



Elem 1 Elem 2 Elem 3

- Row-reverse

```
#conteneur {  
  flex-direction: row-reverse;  
}
```



Elem 3 Elem 2 Elem 1

Exemple « flex-direction » (2/2)

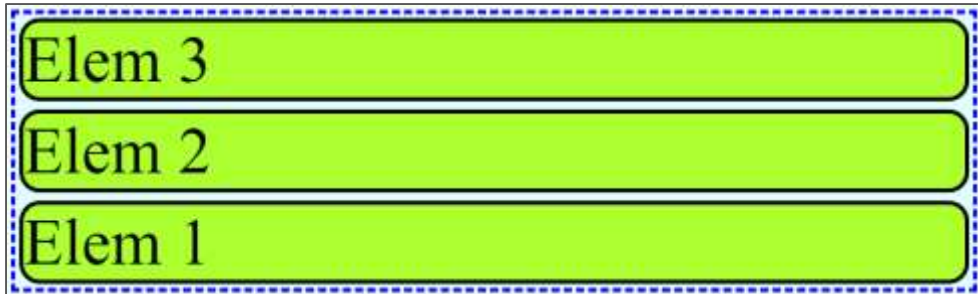
- Column

```
#conteneur {  
    flex-direction: column;  
}
```



- Column-reverse

```
#conteneur {  
    flex-direction: column-reverse;  
}
```



Conteneur flexible sur plusieurs lignes

Passage à la ligne / colonne

Il est possible d'organiser les éléments flexibles pour que ceux-ci se placent sur plusieurs lignes ou colonnes dans le conteneur.

Pour cela, on utilise la propriété « flex-wrap » sur le conteneur avec les valeurs :

- wrap
- wrap-reverse
- nowrap

Par défaut, la valeur de la propriété vaut « no-wrap ».

Passage à la ligne / colonne

- warp / wrap-reverse

Lorsque la taille des éléments dépasse celle du conteneur, les éléments se placeront sur plusieurs lignes ou colonnes.

- nowrap

Les éléments sont rétrécis pour pouvoir tenir sur une seule ligne. Si les éléments ne peuvent plus être rétrécis (ou qu'ils sont non-redimensionnables), cela causera un dépassement du conteneur.

Exemple « flex-wrap » (1/2)

Pour ces exemples, le conteneur et les éléments ont une taille à respecter.

<pre>#conteneur { width: 250px; }</pre>	<pre>.element { min-width: 100px; }</pre>
---	---

- Nowarp

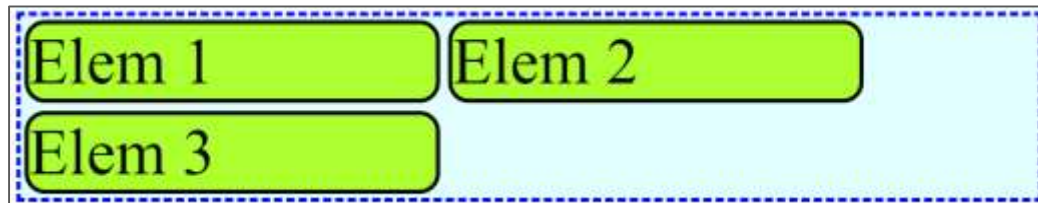
```
#conteneur {  
  flex-wrap: nowrap;  
}
```



Exemple « flex-wrap » (2/2)

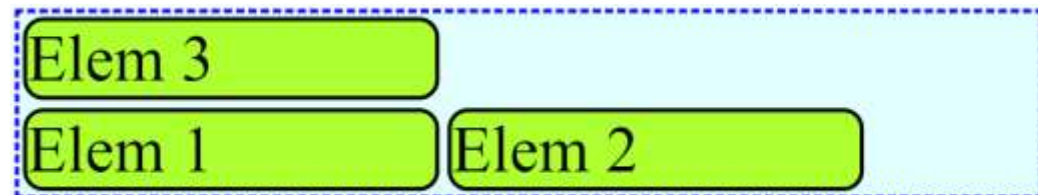
- Warp

```
#conteneur {  
    flex-wrap: wrap;  
}
```



- Wrap-reverse

```
#conteneur {  
    flex-wrap: wrap-reverse;  
}
```



Propriété en syntaxe courte

Il est possible de synthétiser les propriétés « flex-direction » et « flex-wrap » en une seule propriété raccourcie « flex-flow ».

Pour cela, il faut utiliser la propriété avec la valeur de la « flex-direction » suivie de la valeur pour le « flex-wrap ».

```
#conteneur {  
    display: flex;  
    flex-flow: row wrap;  
}
```

Aligner les éléments

Alignement dans l'axe principal

Pour modifier l'alignement dans la direction du « flex-direction », on doit utiliser la propriété « justify-content » sur le conteneur. Celle-ci peut avoir comme valeur :

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

Par défaut, la valeur de la propriété vaut « flex-start ».

Exemple « justify-content » (1/3)

- Flex-start

```
#conteneur {  
  flex-direction: row;  
  justify-content: flex-start;  
}
```



- Flex-end

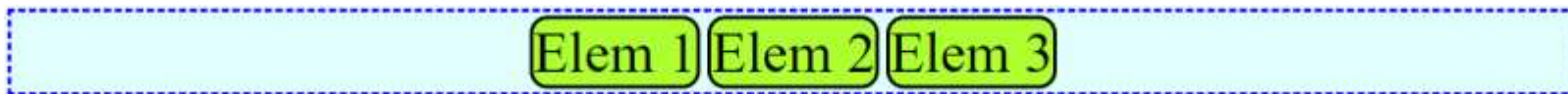
```
#conteneur {  
  flex-direction: row;  
  justify-content: flex-end;  
}
```



Exemple « justify-content » (2/3)

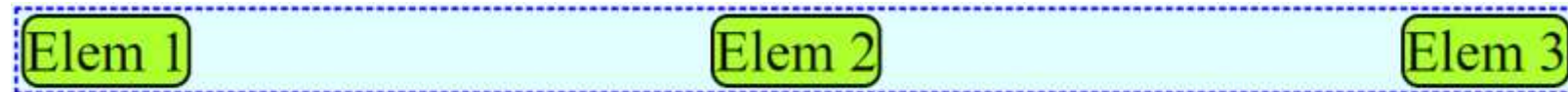
- Center

```
#conteneur {  
  flex-direction: row;  
  justify-content: center;  
}
```



- Space-between

```
#conteneur {  
  flex-direction: row;  
  justify-content: space-between;  
}
```



Exemple « justify-content » (3/3)

- Space-around

```
#conteneur {  
  flex-direction: row;  
  justify-content: space-around;  
}
```



Alignement dans l'axe secondaire

Pour modifier l'alignement dans l'axe secondaire, on doit utiliser la propriété « align-items » sur le conteneur. Celle-ci peut avoir comme valeur :

- flex-start
- flex-end
- stretch
- center
- baseline

Par défaut, la valeur de la propriété est « stretch »

Exemple « align-items » (1/3)

Pour ces exemples, le conteneur a une hauteur définie.

```
#conteneur {  
    height: 40px;  
}
```

- Stretch

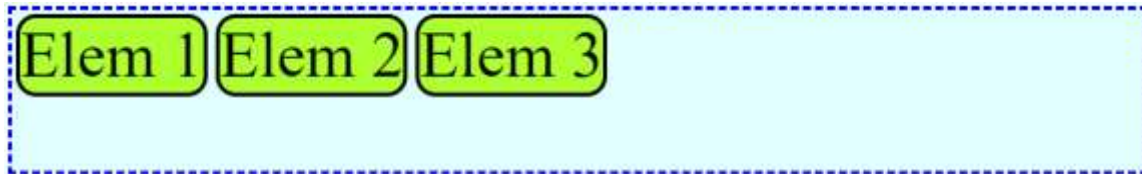
```
#conteneur {  
    align-items: stretch;  
}
```



Exemple « align-items » (2/3)

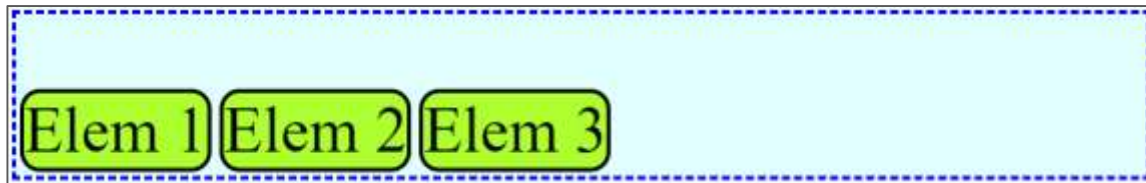
- Flex-start

```
#conteneur {  
  align-items: flex-start;  
}
```



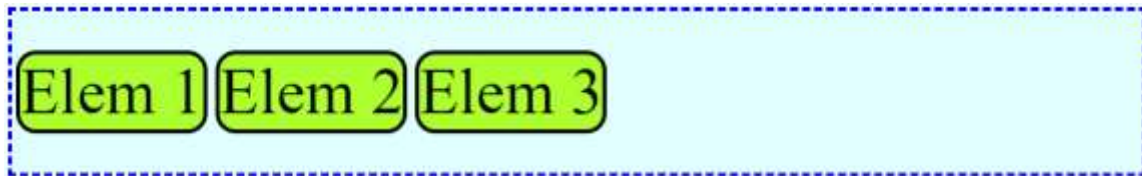
- Flex-end

```
#conteneur {  
  align-items: flex-end;  
}
```



- Center

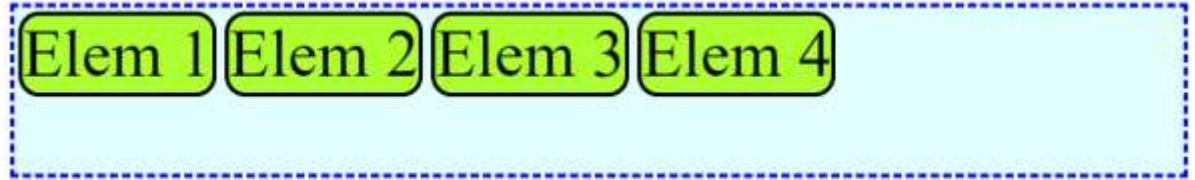
```
#conteneur {  
  align-items: center;  
}
```



Exemple « align-items » (3/3)

- Baseline - Avec des éléments de tailles identiques

```
#conteneur {  
  align-items: baseline;  
}
```



- Baseline - Avec des éléments de tailles différentes

```
#elem2 {  
  font-size: 0.5em;  
}  
  
#elem3 {  
  font-size: 1.5em;  
}
```



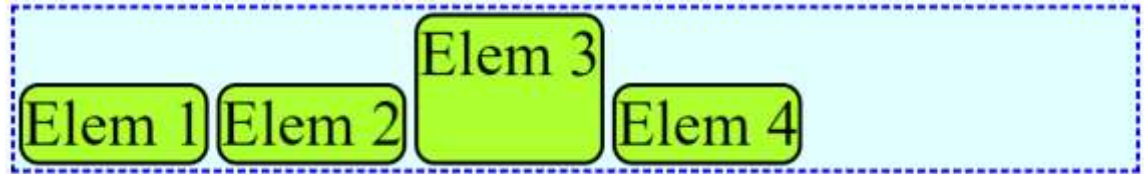
Exception d'alignement pour un élément

Il est possible d'ajouter des exceptions aux règles d'alignement de l'axe secondaire.

Pour cela, il faut utiliser la propriété « align-self » sur les objets désirés. Cette propriété utilise les même valeurs que « align-items ».

Exemple

```
#conteneur {  
    align-items: flex-end;  
}  
  
#elem3 {  
    align-self: stretch;  
}
```



Alignement dans l'axe secondaire avec « Warp »

Pour réaliser un alignement dans l'axe secondaire lorsque le conteneur permet le passage à la ligne / colonne des éléments (flex-wrap), on doit utiliser la propriété « align-content » sur le conteneur avec les valeurs :

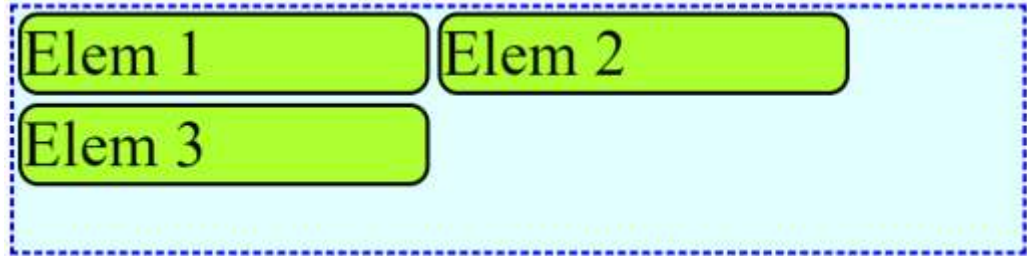
- flex-start
- flex-end
- stretch
- center
- space-between
- space-around

⚠ Attention, pour que la propriété puisse fonctionner, il faut que la hauteur du conteneur soit supérieure à celle qui sera nécessaire pour l'affichage des éléments.

Exemple « align-content » (1/3)

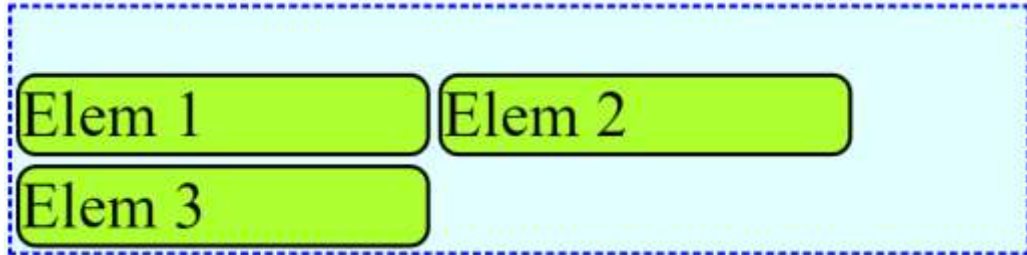
- Flex-start

```
#conteneur {  
  flex-wrap: wrap;  
  align-content: flex-start;  
}
```



- Flex-end

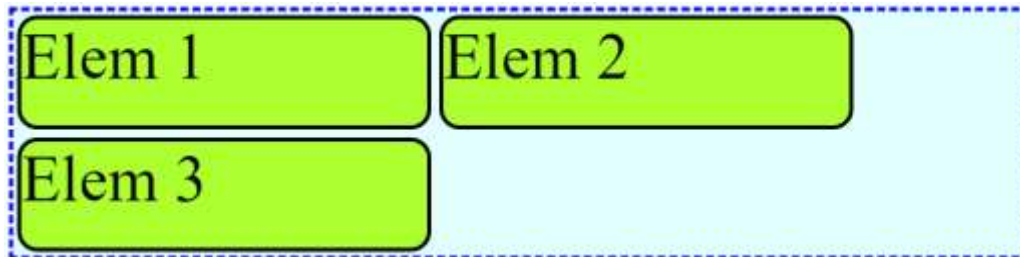
```
#conteneur {  
  flex-wrap: wrap;  
  align-content: flex-end;  
}
```



Exemple « align-content » (2/3)

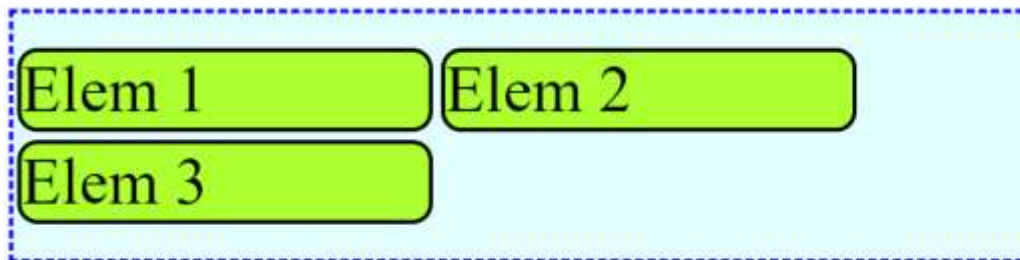
- Stretch

```
#conteneur {  
  flex-wrap: wrap;  
  align-content: stretch;  
}
```



- Center

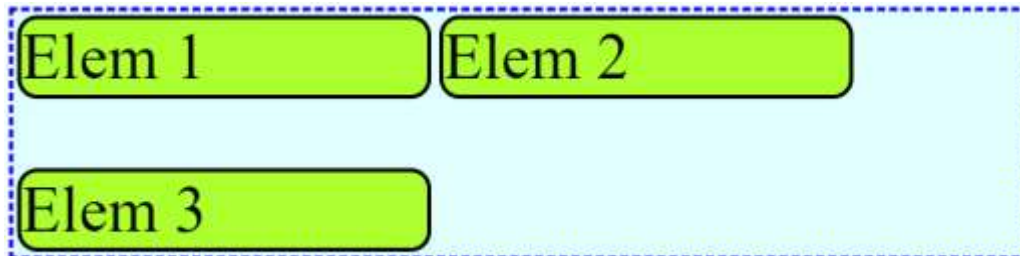
```
#conteneur {  
  flex-wrap: wrap;  
  align-content: center;  
}
```



Exemple « align-content » (3/3)

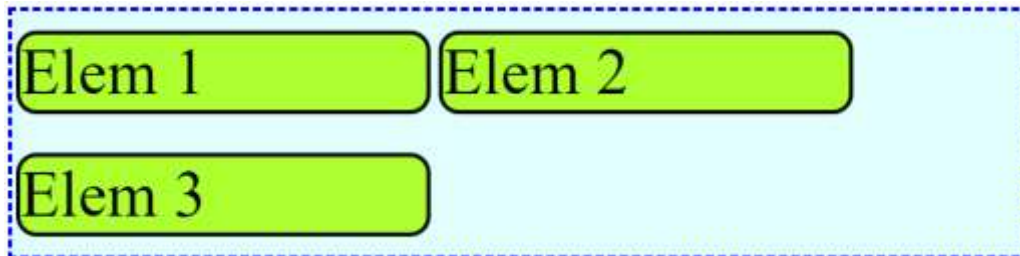
- Space-between

```
#conteneur {  
  flex-wrap: wrap;  
  align-content: space-between;  
}
```



- Space-around

```
#conteneur {  
  flex-wrap: wrap;  
  align-content: space-around;  
}
```



Dimension dynamique des éléments

Contrôler les éléments flexibles

Pour contrôler la dimension des éléments flexibles par rapport à l'espace disponible (dans l'axe principal) du conteneur, on a la possibilité de définir 3 options :

- La taille occupée par l'élément.
- Si l'élément peut être agrandi.
- Si l'élément peut être rétréci.

Définir la taille occupée

Pour définir la taille de base occupée par l'élément, on utilise la propriété « flex-basis ». Celle-ci peut avoir comme valeur :

- auto
- une valeur en « px »

Par défaut, la valeur de la propriété est « auto »

En « auto », le navigateur analyse la taille définie (height ou width) pour l'utiliser. Si aucune valeur n'est définie, il utilisera la taille nécessaire pour afficher le contenu.

Agrandissement des éléments

Pour définir si un élément peut être agrandi, on utilise la propriété « flex-grow ».
Celle-ci attend une valeur entière positive.

- Si la valeur vaut « 0 », l'élément ne peut pas être agrandi.
- Si l'élément a une valeur supérieure à zéro, l'élément peut être agrandi.
Plus la valeur est élevée, plus l'élément prend de la place (quand c'est possible).

Pour diviser l'espace de façon égale, on donne une valeur identique aux éléments.

⚠ L'agrandissement peut être limité par la taille maximale de l'élément.

Par défaut, la valeur de la propriété est « 0 »

Exemple « flex-grow »

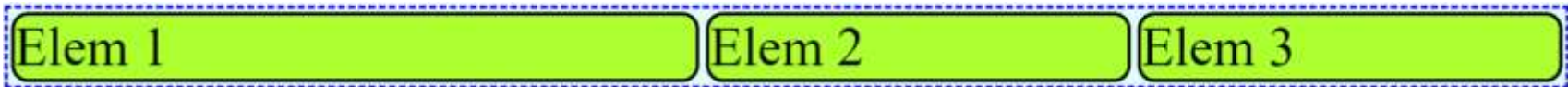
Seul le deuxième élément a une flex-grow de 1



Tous les éléments ont une valeur de flex-grow identique



Le premier élément a un flex-grow égale à 2 et les autres ont un flex-grow de 1



Rétrécissement des éléments

Pour définir si un élément peut être rétréci, on utilise la propriété « flex-shrink ». Celle-ci attend une valeur entière positive.

- Si la valeur vaut « 0 », l'élément ne peut pas être rétréci.
- Si l'élément a une valeur supérieure à zéro, l'élément peut être rétréci.
Plus la valeur est élevée, plus l'élément se compresse si nécessaire.

⚠ Le rétrécissement peut être limité par la taille minimale de l'élément.

Par défaut, la valeur de la propriété est « 1 »

Exemple « flex-shrink »

Pour ces exemples, tous les éléments auront une taille définie à l'aide de « flex-basis » et la taille du conteneur sera réduite.

Etat initial (tous les éléments ont un flex-shrink égal à 1 et une taille identique)



La taille du conteneur est réduite de moitié



Exemple récapitulatif - Définition

Pour ce récapitulatif, nous allons combiner toutes les propriétés ensemble.
L'exemple se compose de 4 éléments qui ont comme propriétés :

```
#elem1 {  
  flex-basis : 100px;  
  flex-grow : 0;  
  flex-shrink: 1;  
  min-width: 50px;  
}
```

```
#elem2 {  
  flex-basis: 100px;  
  flex-grow: 2;  
  flex-shrink: 0;  
}
```

```
#elem3 {  
  flex-basis : 50px;  
  flex-grow : 0;  
  flex-shrink: 0;  
}
```

```
#elem4 {  
  flex-basis : 100px;  
  flex-grow : 1;  
  flex-shrink: 2;  
  min-width: 50px;  
}
```

Exemple de rendu de l'ensemble des éléments



Exemple récapitulatif - Manipulation

État initial



Taille supérieure à la somme des « flex-basis »



Taille cumulé des « flex-basis » atteinte



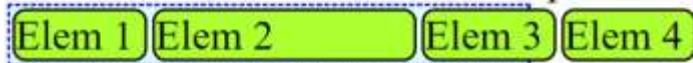
Taille inférieure à la somme des « flex-basis »



Taille limite des éléments atteinte



Taille inférieure au minimum requises!



Propriété en syntaxe courte

Il est possible d'utiliser la propriété raccourcie « flex » pour définir ses options.
Celle-ci fonctionne avec 2 types de valeurs :

→ Les valeurs dans l'ordre : flex-grow, flex-shrink, flex-basis.

→ Des valeurs synthétiques :

- ◆ initial → 0 1 auto
- ◆ auto → 1 1 auto
- ◆ none → 0 0 auto

Modifier l'ordre d'affichage

L'ordre d'affichage

La propriété « order » permet de définir l'ordre d'affichage des éléments. Celle-ci attend une valeur entière (positive ou négative).

Les éléments sont placés dans l'ordre croissant des valeurs de la propriété « order ».

⚠ La propriété « order » est uniquement conçue pour affecter l'ordre visuel ! Elle ne doit pas être utilisée pour modifier l'ordre logique ou l'ordre de tabulation.

Par défaut, la valeur de la propriété est « 0 »

Exemple

Exemple pour 5 éléments, dont l'ordre a été modifié pour 4 d'entre eux.

```
#elem1 {  
    order:5;  
}
```

```
#elem2 {  
    order:1;  
}
```

```
#elem4 {  
    order:-1;  
}
```

```
#elem5 {  
    order:1;  
}
```

Résultat





Merci pour votre attention.

CONSULTING
BSTORM