CSS3 - Flexbox
DU DÉBUTANT AU CONFIRMÉ

Auteur : Paterne Guélablé Gnonzion

Formateur développeur web et mobile : **PHP, Symfony, Rust ...**

Linkedin : Paterne G. G.

Github : teamflp

Procédure pour utiliser ce cours

Bienvenue à ce cours sur le CSS ! Pour tirer le meilleur parti de ce contenu et acquérir une compréhension solide de la matière, suivez les étapes recommandées ci-dessous.

1. **Préparation** :

- Assurez-vous d'avoir les prérequis : connaissance basique du HTML.
- Installez un bon éditeur de code (comme Visual Studio Code).
- Configurez un navigateur moderne pour les tests (Chrome, Firefox, Safari, etc.).

2. **Étude systématique** :

- Commencez par l'**introduction** pour comprendre le contexte du CSS.
- Poursuivez avec les **bases** avant de passer aux concepts avancés.
- Ne sautez pas de sections. Chaque partie est conçue pour s'appuyer sur la précédente.

3. **Pratiquez régulièrement** :

- Après chaque section, prenez le temps de coder et de tester ce que vous avez appris.
- Utilisez les exemples fournis et essayez de les modifier ou de les améliorer.

4. **Participez à des discussions** :

- Rejoignez des forums ou des groupes dédiés au CSS pour poser des questions, partager vos connaissances ou aider d'autres apprenants.
- Partager vos réalisations ou vos défis vous permettra d'obtenir des retours constructifs.

5. **Consultez des ressources supplémentaires** :

- Explorez les **ressources et outils** mentionnés à la fin du cours pour approfondir vos connaissances et améliorer vos compétences.

6. **Mettez en pratique** :

- Essayez de créer un petit projet web à partir de zéro, en appliquant tout ce que vous avez appris.
- Cela renforcera votre compréhension et vous donnera une idée des domaines dans lesquels vous pourriez avoir besoin de plus d'entraînement.

7. **Revenez régulièrement** :

- La technologie et les normes évoluent. Revenez périodiquement pour mettre à jour vos connaissances ou rafraîchir certains concepts.

8. **Évaluez vos progrès** :

- Testez régulièrement vos connaissances. Cela peut être à travers des quiz en ligne, des défis de codage ou des revues de code avec des pairs.

Bonne étude ! Rappelez-vous que l'apprentissage est un voyage, et chaque étape vous rapproche de la maîtrise du CSS.

Introduction à Flexbox

Qu'est-ce que Flexbox?

Flexbox, ou "Flexible Box Layout", est un modèle de mise en page moderne dans CSS3. Contrairement aux méthodes traditionnelles où on s'appuie souvent sur des flottants et des positionnements, Flexbox facilite la conception de structures de mise en page complexes avec une syntaxe plus simple et prévisible. C'est un mode unidimensionnel, ce qui signifie qu'il traite soit la colonne, soit la ligne à la fois, contrairement à la grille CSS qui traite les deux simultanément.

Flexbox permet d'aligner, centrer, ordonner et répartir l'espace entre les éléments d'un conteneur, même lorsque les tailles de vos éléments sont inconnues ou dynamiques.

Pourquoi utiliser Flexbox?

1. **Flexibilité**: Comme son nom l'indique, Flexbox est incroyablement flexible, ce qui permet de créer des mises en page dynamiques qui s'adaptent à différentes tailles d'écran et résolutions d'appareil.
2. **Moins de hacks et de solutions de contournement**: Avant Flexbox, le centrage vertical, par exemple, nécessitait des astuces et des solutions de contournement. Avec Flexbox, c'est simple et direct.
3. **Mise en page unidimensionnelle**: Idéale pour les composants de l'interface utilisateur et les petites mises en page où vous souhaitez organiser des éléments sur une colonne ou une ligne.
4. **Réorganisation**: Avec les propriétés `order`, vous pouvez facilement réorganiser les éléments sans changer le HTML.
5. **Alignement et justification**: Les propriétés `align-items`, `align-content`, `align-self`, `justify-content` permettent un contrôle précis de l'alignement et de la justification des éléments.
6. **Gestion des espaces**: Avec Flexbox, il est facile de répartir l'espace entre les éléments, même si leurs tailles sont inconnues.
7. **Source de cohérence**: Flexbox est supporté par tous les navigateurs modernes, ce qui en fait une option fiable pour la conception web moderne.

En somme, Flexbox offre une solution plus propre, plus robuste et plus cohérente pour concevoir des mises en page complexes, tout en offrant une meilleure adaptabilité aux différents appareils et tailles d'écran. Pour les développeurs qui ont lutté avec les limitations des modèles de mise en page précédents, Flexbox est un véritable soulagement.

Bases de Flexbox

Conteneur flex (flex container) vs Élément flex (flex item)

Flexbox est basé sur un modèle de conteneur et d'élément. Le conteneur est l'élément parent qui contient les éléments enfants, ou éléments flex. Les éléments flex sont les enfants directs du conteneur flex. Les éléments flex peuvent être des éléments HTML ou des éléments générés par CSS, tels que des pseudo-éléments.

Définir un conteneur flex: `display: flex` ou `display: inline-flex`

Pour définir un conteneur flex, on utilise la propriété `display` avec la valeur `flex` ou `inline-flex`. La valeur `flex` crée un conteneur de type bloc, tandis que la valeur `inline-flex` crée un conteneur de type ligne.

`display: flex` est la valeur par défaut de la propriété `display` pour les éléments `<flex>`.

```
<div class="container">
  <div class="container">
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
  </div>
</div>
```

```
.container {
  display: flex; /* ou inline-flex */
  border: 1px solid black; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `display: flex`

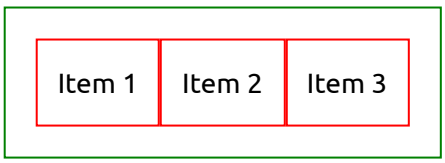
`display: inline-flex` est la valeur par défaut de la propriété `display` pour les éléments `<inline-flex>`.

La propriété `display: inline-flex` fonctionne de la même manière que `display: flex`, sauf que le conteneur lui-même est traité comme un élément en ligne (semblable à un `<span>`) plutôt que comme un bloc (semblable à un `<div>`). Cela signifie qu'il s'alignera avec d'autres éléments en ligne et n'occupera que l'espace nécessaire pour son contenu, au lieu de prendre toute la largeur de son conteneur parent comme le ferait un élément de type block.

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

```
.container {
  display: inline-flex;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid green; /* Juste pour visualiser les éléments */
}
```

Résultat: `display: inline-flex`



Direction et ordre

`flex-direction`: row, row-reverse, column, column-reverse

La propriété `flex-direction` définit la direction principale dans laquelle les éléments flex sont affichés dans le conteneur flex. Il existe quatre valeurs possibles pour `flex-direction`:

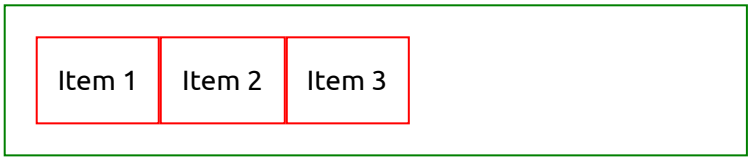
- `row`: Les éléments flex sont affichés horizontalement, de gauche à droite. C'est la valeur par défaut.
- `row-reverse`: Les éléments flex sont affichés horizontalement, de droite à gauche.
- `column`: Les éléments flex sont affichés verticalement, de haut en bas.
- `column-reverse`: Les éléments flex sont affichés verticalement, de bas en haut.

La valeur `row`

```css
.container {
  display: flex;
  flex-direction: row;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}
```


```css
.container {
  display: flex;
  flex-direction: row;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}
```


Résultat: `flex-direction: row`

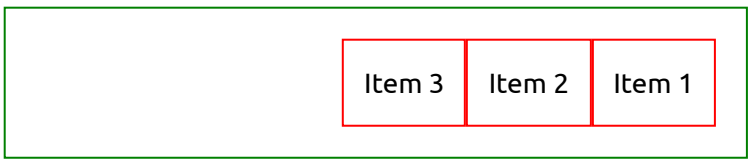


La valeur `row-reverse`

```html
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

```css
.container {
  display: flex;
  flex-direction: row-reverse;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}
```

Résultat: `flex-direction: row-reverse`



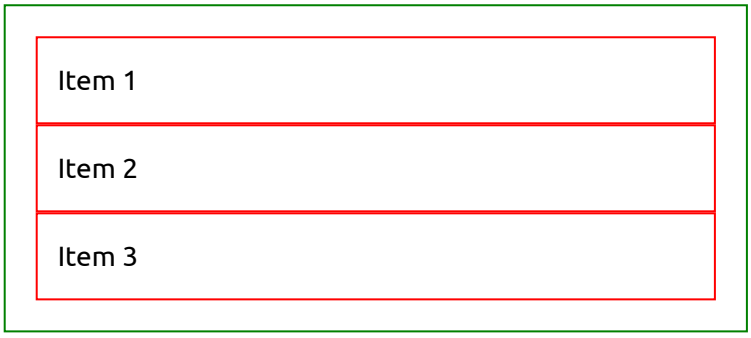
La valeur `column`

```html
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
```

```
  <div class="item">Item 3</div>
</div>
```

```
.container {
  display: flex;
  flex-direction: column;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}
```

Résultat: `flex-direction: column`

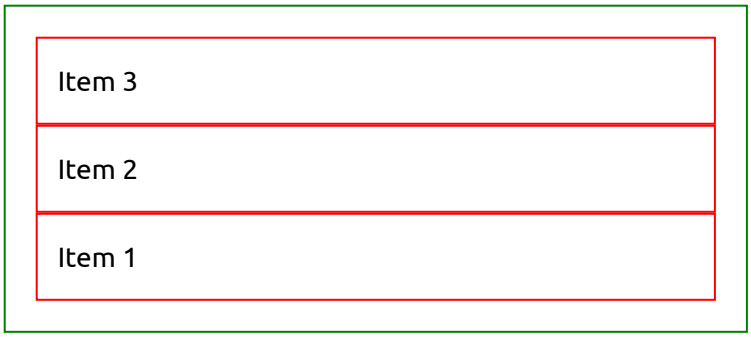


La valeur `column-reverse`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

```
.container {
  display: flex;
  flex-direction: column-reverse;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}
```

Résultat: `flex-direction: column-reverse`



`flex-wrap`: nowrap, wrap, wrap-reverse

La propriété `flex-wrap` définit si les éléments flex doivent être affichés sur une seule ligne ou plusieurs lignes. Il existe trois valeurs possibles pour `flex-wrap`:

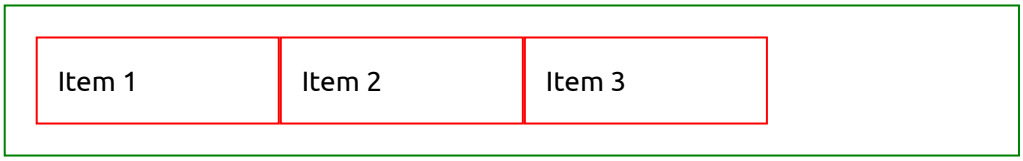
- `nowrap`: Les éléments flex sont affichés sur une seule ligne (ce qui est la valeur par défaut).
- `wrap`: Les éléments flex sont affichés sur plusieurs lignes, de haut en bas.
- `wrap-reverse`: Les éléments flex sont affichés sur plusieurs lignes, de bas en haut.

La valeur `nowrap`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

```
.container {
  display: flex;
  flex-wrap: nowrap;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}
```

Résultat: `flex-wrap: nowrap`

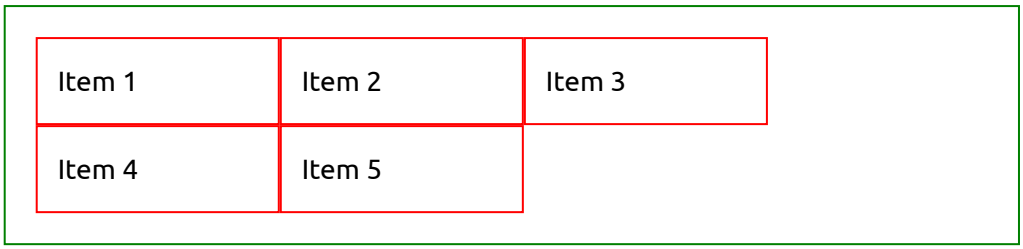


La valeur `wrap`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  flex-wrap: wrap;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}
```

Résultat: `flex-wrap: wrap`

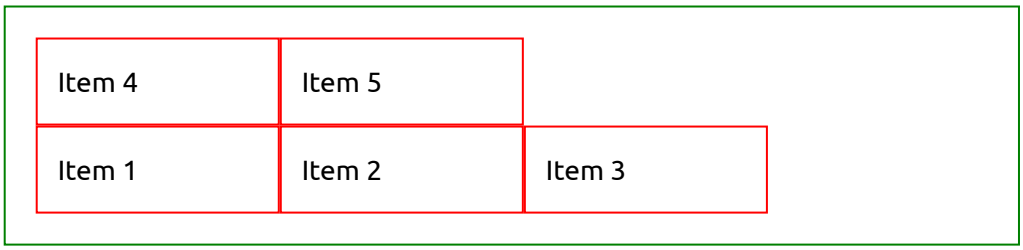


La valeur `wrap-reverse`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  flex-wrap: wrap-reverse;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}
```

Résultat: `flex-wrap: wrap-reverse`



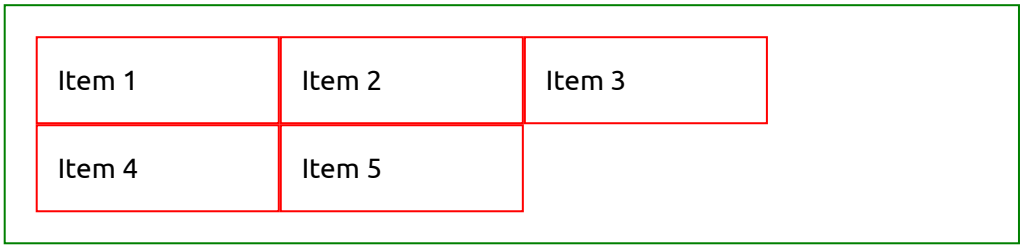
`flex-flow`: combinaison de flex-direction et flex-wrap

La propriété `flex-flow` est une propriété raccourcie qui combine les propriétés `flex-direction` et `flex-wrap`. La valeur par défaut de `flex-flow` est `row nowrap`.

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  flex-flow: row wrap;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}
```

Résultat: `flex-flow: row wrap`



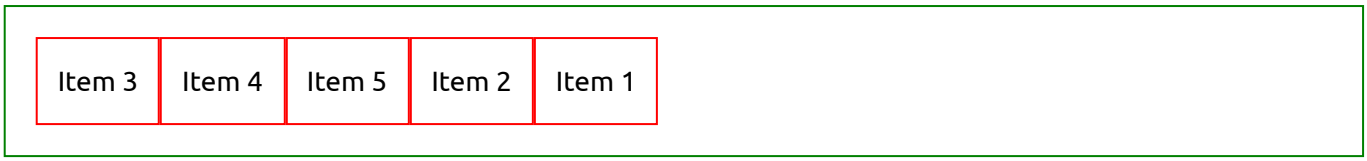
`order`: définir l'ordre des éléments flex

La propriété `order` définit l'ordre dans lequel les éléments flex sont affichés dans le conteneur flex. La valeur par défaut de `order` est `0`. Les valeurs négatives sont autorisées.

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}

.item:nth-child(1) {
  order: 2;
}

.item:nth-child(2) {
  order: 1;
}
```

Résultat: `order: 1`



Avec cette configuration, `Item 2` apparaîtra avant `Item 1` dans le conteneur.

Alignement et justification

`justify-content`: flex-start, flex-end, center, space-between, space-around, space-evenly

La propriété `justify-content` définit l'alignement des éléments flex le long de l'axe principal (main axis). Il existe six valeurs possibles pour `justify-content`:

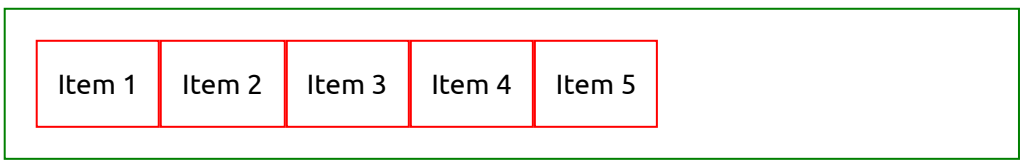
- `flex-start`: Les éléments flex sont alignés au début du conteneur flex. C'est la valeur par défaut.
- `flex-end`: Les éléments flex sont alignés à la fin du conteneur flex.
- `center`: Les éléments flex sont centrés dans le conteneur flex.
- `space-between`:
- `space-around`:
- `space-evenly`:

- La valeur `flex-start`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  justify-content: flex-start;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `justify-content: flex-start`



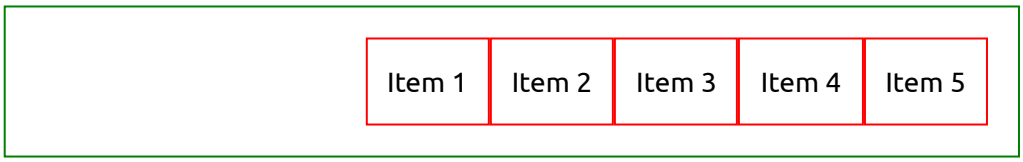
Avec cette configuration, tous les éléments flex seront alignés au début du conteneur flex.

- La valeur `flex-end`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  justify-content: flex-end;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `justify-content: flex-end`



Avec cette configuration, tous les éléments flex seront alignés à la fin du conteneur flex.

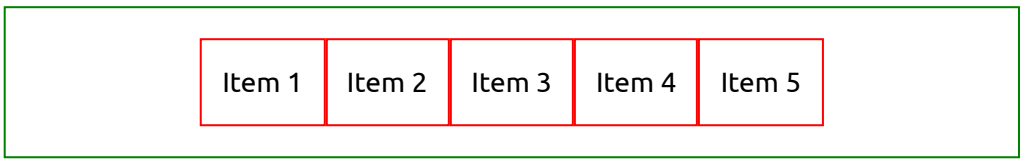
- La valeur `center`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  justify-content: center;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
```

```
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `justify-content: center`



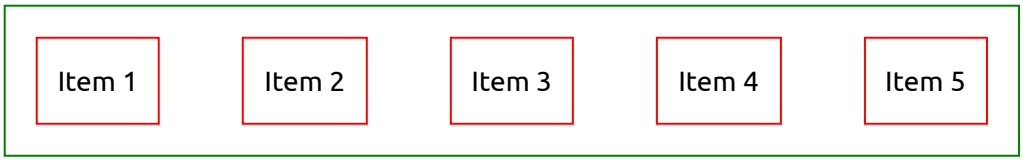
Avec cette configuration, tous les éléments flex seront centrés dans le conteneur flex.

- La valeur `space-between`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  justify-content: space-between;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `justify-content: space-between`



Dans cet exemple, vous verrez que le premier élément est aligné avec le début du conteneur, le dernier élément est aligné avec la fin du conteneur, et l'espace restant est réparti également entre les éléments.

Il faut savoir que si vous n'avez que deux éléments dans votre conteneur flex, space-between aura le même effet que si vous aviez utilisé `flex-start` pour le premier élément et `flex-end` pour le second.

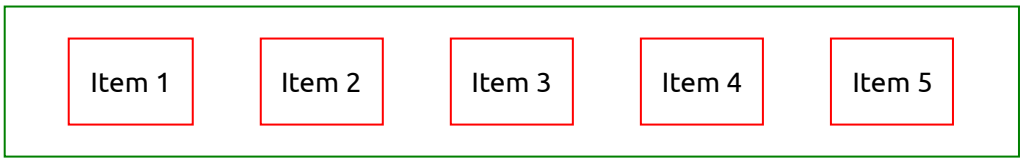
- La valeur `space-around`

Lorsque vous utilisez la valeur space-around pour la propriété justify-content, chaque élément flex reçoit un espace égal de part et d'autre. Cela signifie que l'espace entre deux éléments adjacents sera le double de l'espace entre l'un de ces éléments et le bord du conteneur. En d'autres termes, les éléments flex bénéficient d'un espace égal avant et après eux.

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  justify-content: space-around;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `justify-content: space-around`



Dans cet exemple, l'espace autour de chaque élément est réparti de manière égale. Par conséquent, l'espace entre deux éléments consécutifs sera toujours égal au double de l'espace entre un élément et le bord du conteneur.

C'est une propriété utile lorsque vous voulez que vos éléments soient répartis uniformément, tout en ayant une certaine marge autour d'eux par rapport aux bords du conteneur.

- La valeur `space-evenly`

La valeur space-evenly pour la propriété `justify-content` répartit uniformément l'espace entre les éléments flex et aussi entre les éléments et les bords du conteneur. C'est une distribution très équilibrée de l'espace, différente de `space-around` et `space-between`.

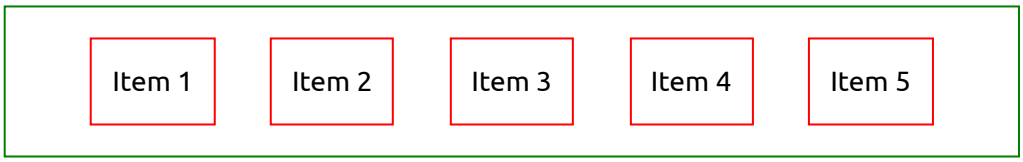
En utilisant `space-evenly`, l'espace entre deux éléments adjacents est le même que l'espace entre le premier élément et le bord du conteneur, ainsi que l'espace entre le dernier élément et le bord du

conteneur.

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  justify-content: space-evenly;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `justify-content: space-evenly`



Avec cette configuration, l'espace est réparti de la manière la plus uniforme possible, tant entre les éléments eux-mêmes qu'entre les éléments et les bords du conteneur. C'est une excellente option si vous souhaitez une distribution totalement égale de l'espace dans votre conteneur flex.

`align-items`: flex-start, flex-end, center, baseline, stretch

La propriété `align-items` est utilisée en CSS Flexbox pour aligner les éléments flex le long de l'axe transversal (cross axis), c'est-à-dire perpendiculairement à l'axe principal. Cette propriété a plusieurs valeurs possibles :

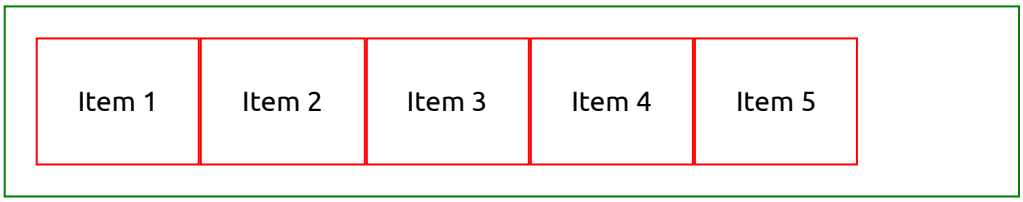
- `flex-start`: Les éléments flex sont alignés au début du conteneur flex. C'est la valeur par défaut.
- `flex-end`: Les éléments flex sont alignés à la fin du conteneur flex.
- `center`: Les éléments flex sont centrés dans le conteneur flex.
- `baseline`: Les éléments flex sont alignés sur leur ligne de base commune. Cela signifie que les éléments flex sont alignés en fonction de leur texte.
- `stretch`: Les éléments flex sont étirés pour remplir le conteneur flex.

- La valeur `flex-start`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  align-items: flex-start;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `align-items: flex-start`



Avec cette configuration, tous les éléments flex seront alignés au haut du conteneur flex.

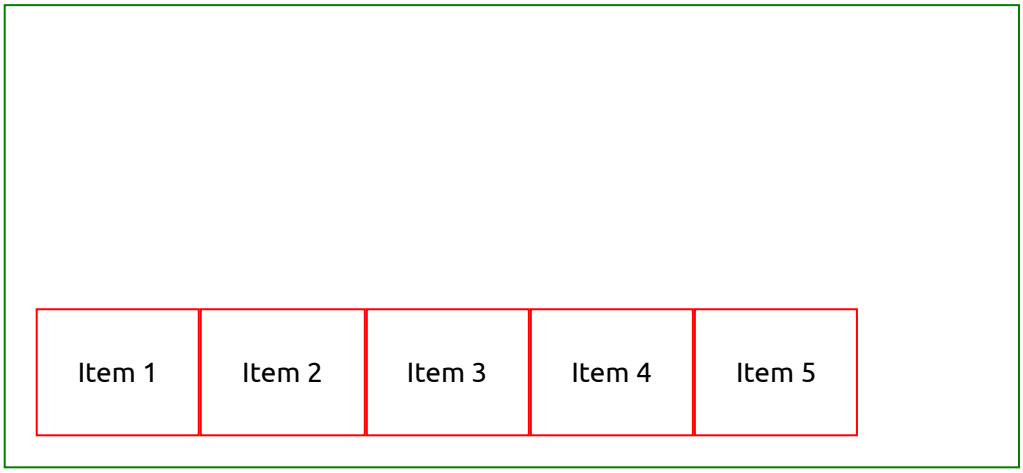
- La valeur `flex-end`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  align-items: flex-end;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
```

```
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `align-items: flex-end`



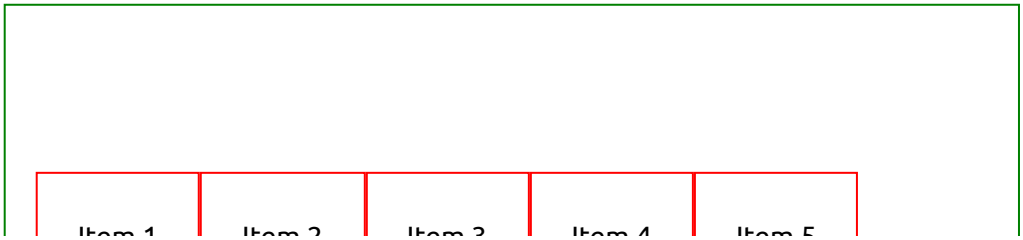
Avec cette configuration, tous les éléments flex seront alignés au bas du conteneur flex.

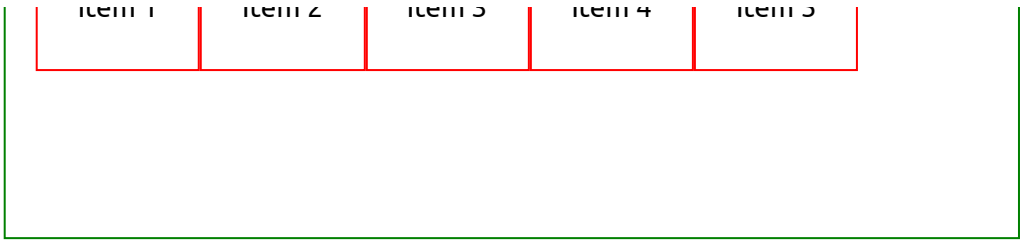
- La valeur `center`

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
</div>
```

```
.container {
  display: flex;
  align-items: center;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `align-items: center`





Avec cette configuration, tous les éléments flex seront alignés au centre du conteneur flex.

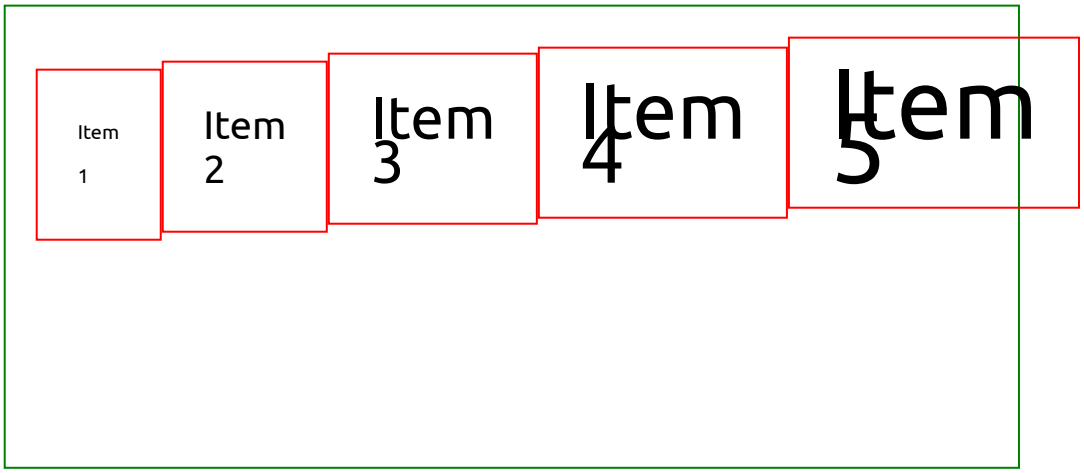
- La valeur `baseline`

La valeur `baseline` pour la propriété `align-items` aligne les éléments flex sur leur ligne de base commune. Cela signifie que les éléments flex sont alignés en fonction de leur texte.

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item" style="font-size: 20px;">Item 2</div>
  <div class="item" style="font-size: 30px;">Item 3</div>
  <div class="item" style="font-size: 40px;">Item 4</div>
  <div class="item" style="font-size: 50px;">Item 5</div>
</div>
```

```
.container {
  display: flex;
  align-items: baseline;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 20px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `align-items: baseline`



Avec cette configuration, tous les éléments flex seront alignés sur leur ligne de base commune. Cela signifie que les éléments flex sont alignés en fonction de leur texte.

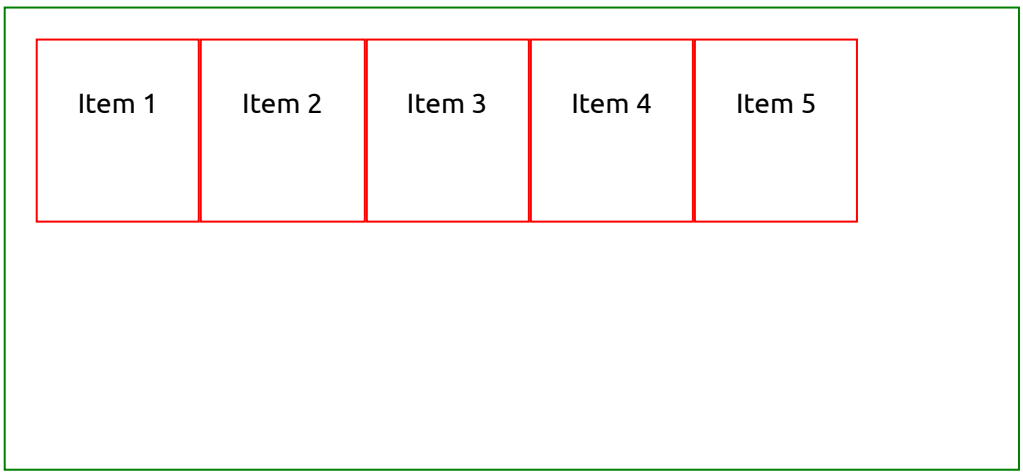
- La valeur `stretch`

La valeur `stretch` pour la propriété `align-items` étire les éléments flex pour remplir le conteneur flex. C'est la valeur par défaut de `align-items`.

```
<div class="container">
  <div class="item" style="height: 50px;">Item 1</div>
  <div class="item" style="height: 50px;">Item 2</div>
  <div class="item" style="height: 50px;">Item 3</div>
  <div class="item" style="height: 50px;">Item 4</div>
  <div class="item" style="height: 50px;">Item 5</div>
</div>
```

```
.container {
  display: flex;
  align-items: stretch;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 20px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `align-items: stretch`



Avec cette configuration, tous les éléments flex seront étirés pour remplir le conteneur flex.

`align-content`: flex-start, flex-end, center, space-between, space-around, stretch

La propriété `align-content` définit l'alignement des lignes de l'axe transversal (cross axis) dans un conteneur flex. Cette propriété a plusieurs valeurs possibles :

- `flex-start`: Les lignes sont alignées au début du conteneur flex. C'est la valeur par défaut.
- `flex-end`: Les lignes sont alignées à la fin du conteneur flex.
- `center`: Les lignes sont centrées dans le conteneur flex.
- `space-between`: Les lignes sont réparties uniformément dans le conteneur flex. L'espace entre deux lignes consécutives est le même que l'espace entre le premier et le dernier élément.
- `space-around`: Les lignes sont réparties uniformément dans le conteneur flex. L'espace entre deux lignes consécutives est le double de l'espace entre le premier et le dernier élément.
- `stretch`: Les lignes sont étirées pour remplir le conteneur flex.

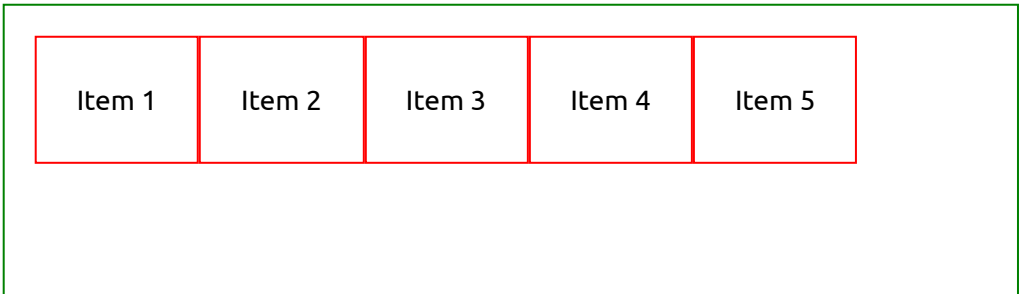
- La valeur `flex-start`

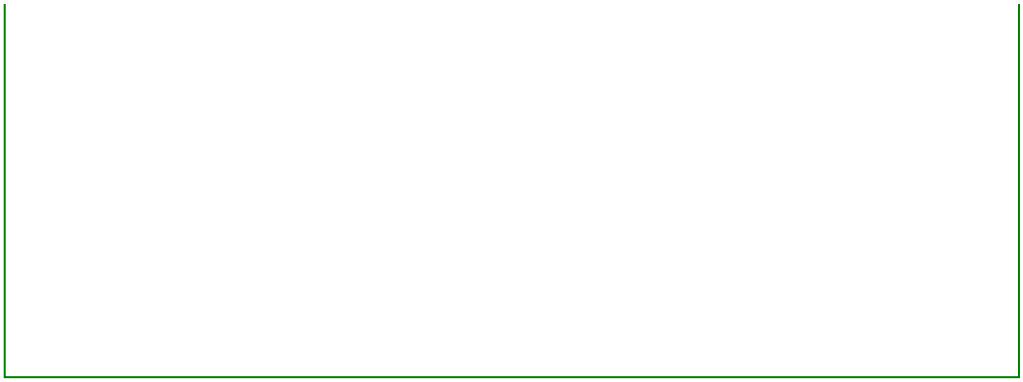
La propriété align-content détermine la manière dont les lignes d'un conteneur flex sont réparties lorsque l'espace restant sur l'axe transversal (cross axis) est supérieur à celui requis pour afficher ces lignes. Elle est surtout pertinente lorsqu'il y a plusieurs lignes d'éléments, c'est-à-dire lorsque flex-wrap est réglé sur wrap.

```
<div class="container">
  <div class="item" style="height: 50px;">Item 1</div>
  <div class="item" style="height: 50px;">Item 2</div>
  <div class="item" style="height: 50px;">Item 3</div>
  <div class="item" style="height: 50px;">Item 4</div>
  <div class="item" style="height: 50px;">Item 5</div>
</div>
```

```
.container {
  display: flex;
  flex-wrap: wrap;
  align-content: flex-end;
  height: 300px; /* Hauteur fixe pour bien montrer l'effet de align-content
*/
  width: 70%;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  /* Permet de forcer l'enveloppement des éléments */
  padding: 20px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `align-content: flex-start`





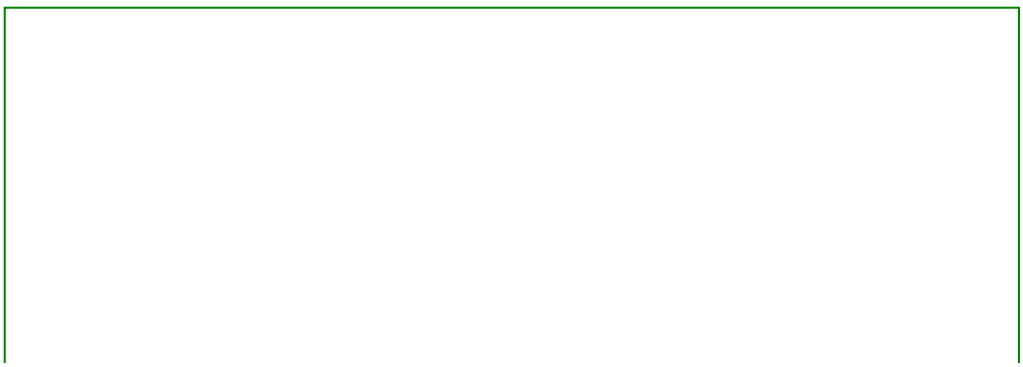
Les lignes sont emballées vers le haut du conteneur. L'espace excédentaire est placé sous la dernière ligne.

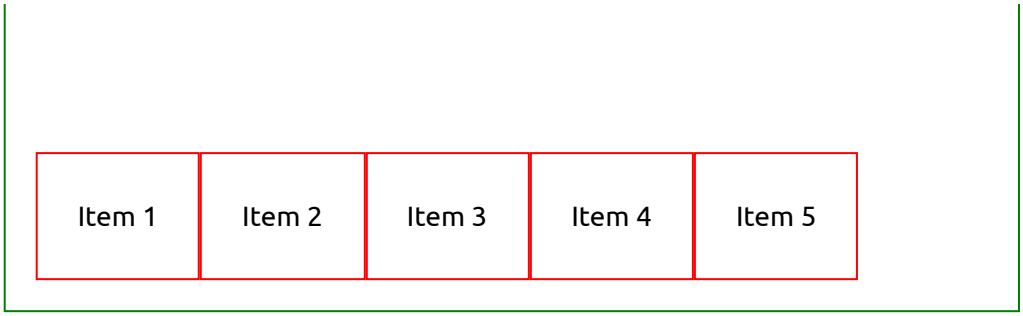
- La valeur `flex-end`

```
<div class="container">
  <div class="item" style="height: 50px;">Item 1</div>
  <div class="item" style="height: 50px;">Item 2</div>
  <div class="item" style="height: 50px;">Item 3</div>
  <div class="item" style="height: 50px;">Item 4</div>
  <div class="item" style="height: 50px;">Item 5</div>
</div>
```

```
.container {
  display: flex;
  flex-wrap: wrap;
  align-content: flex-end;
  height: 300px; /* Hauteur fixe pour bien montrer l'effet de align-content
*/
  width: 70%;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  /* Permet de forcer l'enveloppement des éléments */
  padding: 20px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `align-content: flex-end`





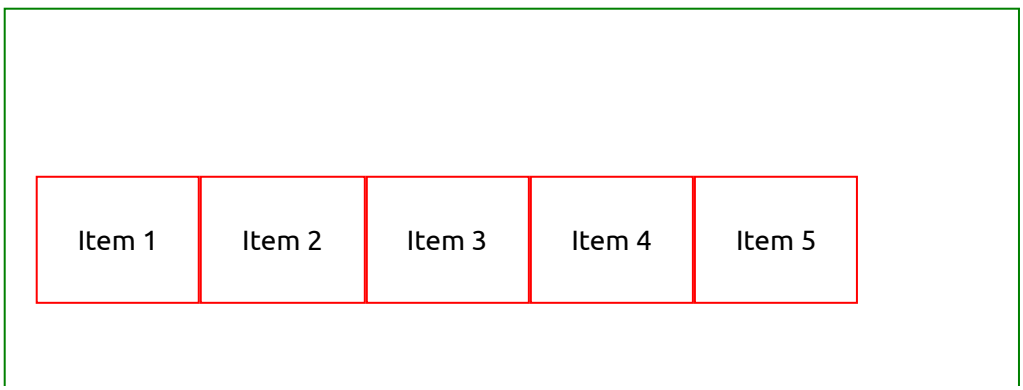
Avec cette configuration, les éléments s'envelopperont (grâce à flex-wrap: wrap) et l'espace excédentaire sera placé au-dessus des lignes d'éléments (grâce à align-content: flex-end).

- La valeur `center`

```
<div class="container">
  <div class="item" style="height: 50px;">Item 1</div>
  <div class="item" style="height: 50px;">Item 2</div>
  <div class="item" style="height: 50px;">Item 3</div>
  <div class="item" style="height: 50px;">Item 4</div>
  <div class="item" style="height: 50px;">Item 5</div>
</div>
```

```
.container {
  display: flex;
  flex-wrap: wrap;
  align-content: center;
  height: 300px; /* Hauteur fixe pour bien montrer l'effet de align-content
*/
  width: 70%;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  /* Permet de forcer l'enveloppement des éléments */
  padding: 20px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `align-content: center`



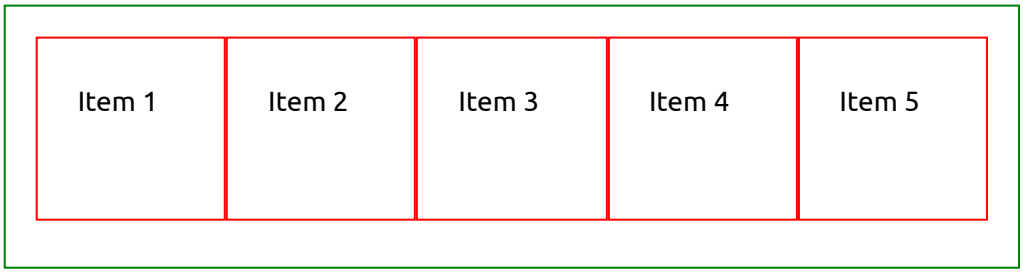
Les lignes sont centrées verticalement dans le conteneur, avec un espace égal au-dessus et en dessous.

- La valeur `space-between`

```
<div class="container">
  <div class="item" style="height: 50px;">Item 1</div>
  <div class="item" style="height: 50px;">Item 2</div>
  <div class="item" style="height: 50px;">Item 3</div>
  <div class="item" style="height: 50px;">Item 4</div>
  <div class="item" style="height: 50px;">Item 5</div>
</div>
```

```
.container {
  display: flex;
  flex-wrap: wrap;
  align-content: space-between;
  height: 100px; /* Hauteur fixe pour bien montrer l'effet de align-content
*/
  width: 70%;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  width: 30%; /* Permet de forcer l'enveloppement des éléments */
  padding: 20px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `align-content: space-between`



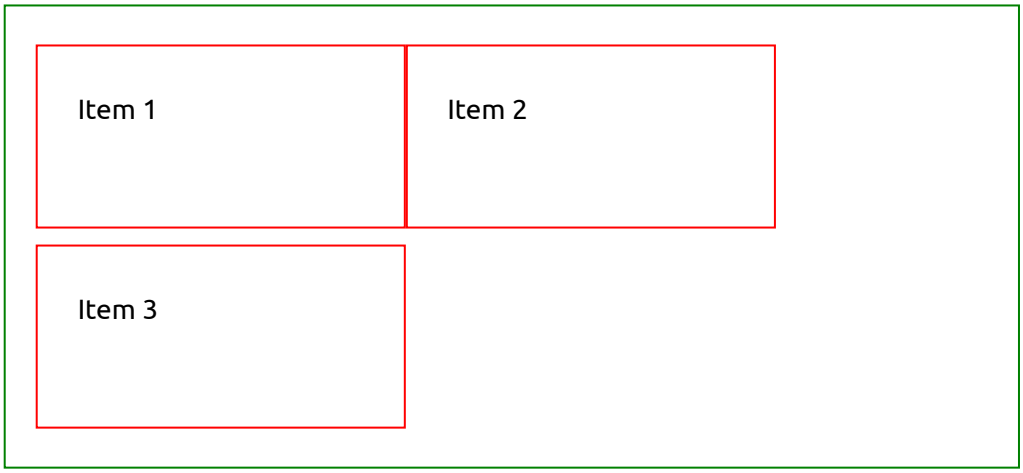
Lorsque plusieurs lignes sont présentes dans un conteneur flex, la propriété align-content: space-between répartit uniformément l'espace entre les lignes. Cela signifie qu'il n'y a pas d'espace excédentaire au-dessus de la première ligne ni en dessous de la dernière ligne, mais l'espace est réparti également entre les lignes intermédiaires.

- La valeur `space-around`

```
<div class="container">
  <div class="item" style="height: 50px;">Item 1</div>
  <div class="item" style="height: 50px;">Item 2</div>
  <div class="item" style="height: 50px;">Item 3</div>
</div>
```

```
.container {
  display: flex;
  flex-wrap: wrap;
  align-content: space-around;
  height: 100px; /* Hauteur fixe pour bien montrer l'effet de align-content
*/
  width: 70%;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  width: 30%; /* Permet de forcer l'enveloppement des éléments */
  padding: 20px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `align-content: space-around`



Lorsque plusieurs lignes sont présentes dans un conteneur flex, la propriété align-content: space-around répartit uniformément l'espace entre les lignes. Cela signifie qu'il n'y a pas d'espace excédentaire au-dessus de la première ligne ni en dessous de la dernière ligne, mais l'espace est réparti également entre les lignes intermédiaires.

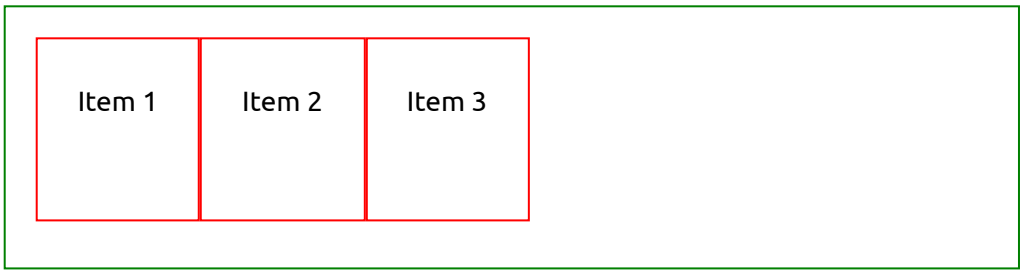
- La valeur `stretch`

```
<div class="container">
  <div class="item" style="height: 50px;">Item 1</div>
  <div class="item" style="height: 50px;">Item 2</div>
```

```
  <div class="item" style="height: 50px;">Item 3</div>
</div>
```

```
.container {
  display: flex;
  flex-wrap: wrap;
  align-content: stretch;
  height: 100px; /* Hauteur fixe pour bien montrer l'effet de align-content
*/
  width: 70%;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}
```

Résultat: `align-content: stretch`



Lorsque plusieurs lignes sont présentes dans un conteneur flex, la propriété align-content: stretch étire les lignes pour remplir le conteneur flex. C'est la valeur par défaut de align-content.

align-self : auto, flex-start, flex-end, center, baseline, stretch

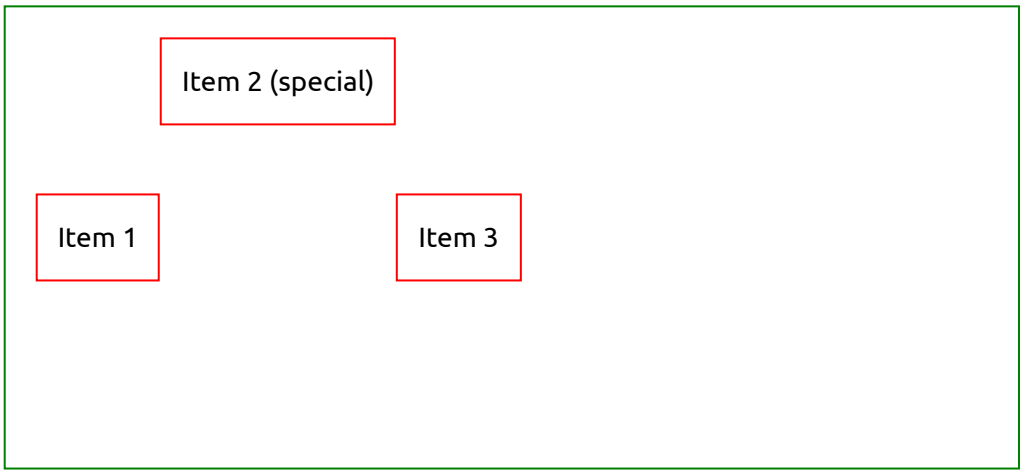
La propriété CSS align-self permet d'ajuster l'alignement d'un élément individuel à l'intérieur d'un conteneur flex ou grid. Cette propriété est très utile lorsque nous voulons spécifiquement changer l'alignement d'un élément particulier sans affecter les autres éléments du conteneur. Elle possède plusieurs valeurs possibles :

- `auto`: La valeur par défaut. L'élément hérite de la valeur de align-items de son parent, ou de stretch si l'élément n'a pas de parent.
- `flex-start`: L'élément est aligné au début du conteneur flex.
- `flex-end`: L'élément est aligné à la fin du conteneur flex.
- `center`: L'élément est centré dans le conteneur flex.
- `baseline`: L'élément est aligné sur sa ligne de base commune. Cela signifie que l'élément est aligné en fonction de son texte.
- `stretch`: L'élément est étiré pour remplir le conteneur flex.

```
<div class="container">
  <div class="item default">Item 1</div>
  <div class="item special">Item 2 (special)</div>
  <div class="item default">Item 3</div>
</div>
```

```css
.container {
  display: flex;
  align-items: center; /* Alignement par défaut pour tous les éléments */
  height: 200px;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}

.special {
  align-self: flex-start; /* Cet élément sera aligné au début du conteneur
*/
}
```

Résultat: `align-self: flex-start`



L'élément "Item 2 (special)" est aligné au début du conteneur grâce à align-self: flex-start, alors que les autres éléments sont centrés à cause de `align-items: center` sur le conteneur.

Taille des éléments flex

Lors de la conception de mises en page flexibles avec Flexbox, il est souvent nécessaire d'ajuster la taille des éléments en fonction de l'espace disponible. Pour cela, Flexbox offre plusieurs propriétés pour contrôler la capacité des éléments à grandir, rétrécir, ou définir leur taille de base.

`flex-grow`: 0, 1, 2, 3, 4, 5 : Définit la capacité d'un élément à grandir

La propriété `flex-grow` définit la capacité d'un élément à grandir pour remplir l'espace disponible dans un conteneur flex. Cette propriété a plusieurs valeurs possibles :

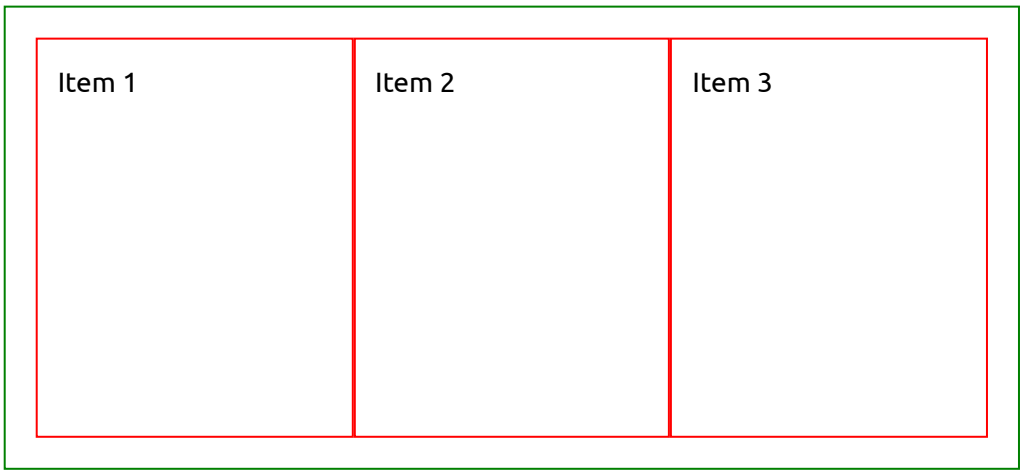
- `0`: L'élément ne grandira pas pour remplir l'espace disponible. C'est la valeur par défaut.
- `1`: L'élément grandira pour remplir l'espace disponible.
- `2`: L'élément grandira deux fois plus que les autres éléments.
- `3`: L'élément grandira trois fois plus que les autres éléments.

- 4: L'élément grandira quatre fois plus que les autres éléments.
- 5: L'élément grandira cinq fois plus que les autres éléments.

```
<div class="container">
  <div class="item grow-1">Item 1</div>
  <div class="item grow-2">Item 2</div>
  <div class="item grow-3">Item 3</div>
</div>
```

```
.container {
  display: flex;
  height: 200px;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}

.grow-1 {
  flex-grow: 1;
}

.grow-2 {
  flex-grow: 2;
}

.grow-3 {
  flex-grow: 3;
}
```

Résultat: `flex-grow: 1`



Avec cette configuration, tous les éléments flex grandiront pour remplir l'espace disponible dans le conteneur flex. Comme tous les éléments ont la même valeur de flex-grow, ils grandiront tous de la même

quantité.

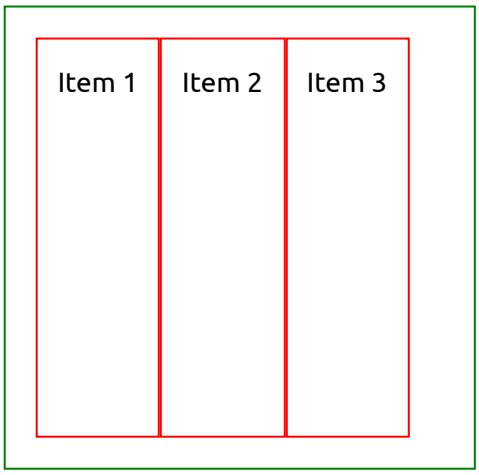
`flex-shrink`: Définit la capacité d'un élément à rétrécir

La propriété `flex-shrink` fonctionne de la même manière que `flex-grow`, mais dans le sens inverse. Elle détermine la capacité d'un élément à rétrécir lorsque l'espace est limité. La valeur par défaut est `1`, ce qui signifie que l'élément se réduira pour éviter un débordement.

```
<div class="container">
  <div class="item shrink-1">Item 1</div>
  <div class="item shrink-2">Item 2</div>
  <div class="item shrink-3">Item 3</div>
</div>
```

```
.container {
  display: flex;
  height: 200px;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  flex-shrink: 1;
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `flex-shrink: 1`



Cela signifie que l'élément a une capacité de croissance (flex-grow) de 1, une capacité de rétrécissement (`flex-shrink`) de `0`, et une taille de base (`flex-basis`) de `200px`.

L'utilisation de ces propriétés permet d'obtenir une grande flexibilité et un contrôle précis sur la manière dont les éléments s'ajustent et se répartissent dans un conteneur flex.

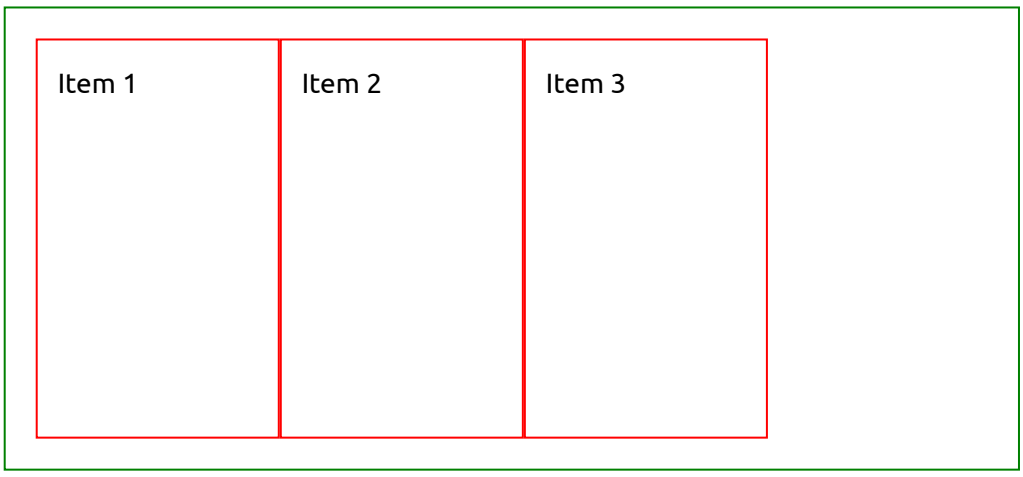
`flex-basis`: Définit la taille de base d'un élément

La propriété `flex-basis` définit la taille de base d'un élément avant qu'il ne grandisse ou ne rétrécisse en fonction des propriétés `flex-grow` et `flex-shrink`. Elle peut accepter des valeurs en `%, px, em, etc.`, ou la valeur `auto`, qui utilise la taille intrinsèque de l'élément ou sa valeur `width` ou `height`.

```
<div class="container">
  <div class="item basis-1">Item 1</div>
  <div class="item basis-2">Item 2</div>
  <div class="item basis-3">Item 3</div>
</div>
```

```
.container {
  display: flex;
  height: 200px;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  flex-basis: 100px;
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `flex-basis: 100px`



Cela signifie que l'élément a une capacité de croissance (flex-grow) de 1, une capacité de rétrécissement (`flex-shrink`) de `1`, et une taille de base (`flex-basis`) de `100px`.

`flex`: raccourci pour flex-grow, flex-shrink et flex-basis

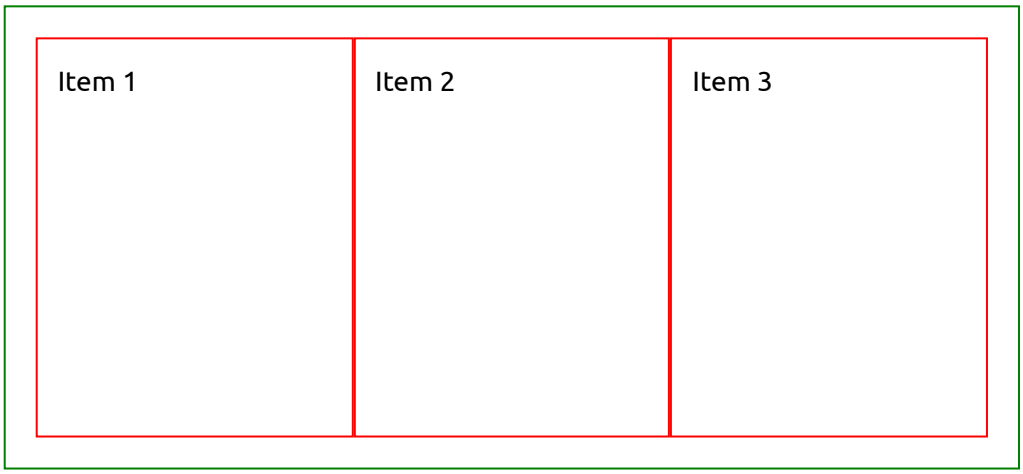
La propriété flex est un raccourci pour définir flex-grow, flex-shrink, et flex-basis en une seule déclaration. La syntaxe générale est : flex: [flex-grow] [flex-shrink] [flex-basis]

```
<div class="container">
  <div class="item flex-1">Item 1</div>
  <div class="item flex-2">Item 2</div>
```

```
  <div class="item flex-3">Item 3</div>
</div>
```

```
.container {
  display: flex;
  height: 200px;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  flex: 1 1 100px;
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `flex: 1 1 100px`



Cela signifie que l'élément a une capacité de croissance (flex-grow) de 1, une capacité de rétrécissement (`flex-shrink`) de `1`, et une taille de base (`flex-basis`) de `100px`.

Autres propriétés

`gap`: Définit l'espace entre les éléments

La propriété `gap` est utilisée pour définir l'espacement entre les éléments `flex` au sein d'un conteneur flex. Elle est particulièrement utile car elle permet d'ajouter de l'espacement entre les éléments sans affecter les marges extérieures du conteneur. Cette propriété était initialement associée aux grilles (`Grid Layout`), mais elle est maintenant applicable aux conteneurs flex (Flexbox) dans les navigateurs modernes.

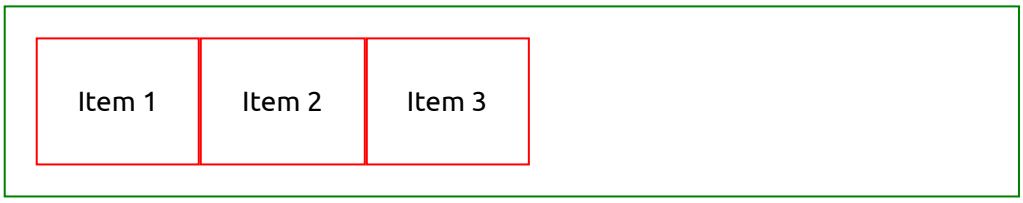
La propriété gap peut accepter une ou deux valeurs :

1. Une seule valeur (par exemple, `gap: 20px;`) : Cette valeur s'appliquera à la fois horizontalement (entre les éléments sur une ligne) et verticalement (entre les lignes).
2. Deux valeurs (par exemple, gap: `20px 30px;`) : La première valeur s'applique verticalement (entre les lignes) et la seconde horizontalement (entre les éléments sur une ligne).

```html
<div class="flex-container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

```css
.flex-container {
  display: flex;
  gap: 20px;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.item {
  padding: 20px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `gap: 20px`



Dans cet exemple, un espacement de `20px` sera ajouté entre chaque élément flex du conteneur.

L'utilisation de `gap` avec Flexbox simplifie la gestion de l'espacement entre les éléments, car il n'est plus nécessaire d'utiliser des marges complexes et des pseudo-éléments pour gérer l'espacement sans affecter les marges extérieures du conteneur.

Exemples pratiques

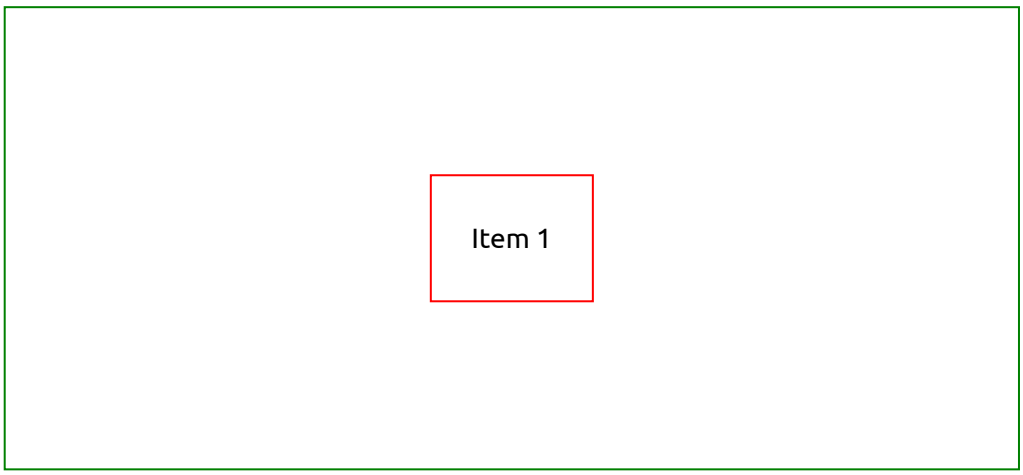
Centrer un élément dans un conteneur flex

```html
<div class="container">
  <div class="item">Item 1</div>
</div>
```

```css
.container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 200px;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}
```

```css
.item {
  padding: 20px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `justify-content: center; align-items: center;`



Mise en page d'en-tête avec logo et navigation

```html
<header class="header">
  <div class="logo">Logo</div>
  <nav class="nav">
    <a href="#">Accueil</a>
    <a href="#">À propos</a>
    <a href="#">Contact</a>
  </nav>
</header>
```

```css
.header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 20px;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

.logo {
  font-size: 1.5rem;
  font-weight: bold;
}

.nav {
  display: flex;
  gap: 20px;
}
```

```css
.nav a {
  text-decoration: none;
  color: black;
}
```

Résultat: `justify-content: space-between; align-items: center;`

Mise en page d'un formulaire

```html
<form action="#" method="post">
  <label for="name">Nom :</label>
  <input type="text" id="name" name="name" />

  <label for="email">E-mail :</label>
  <input type="email" id="email" name="email" />

  <label for="message">Message :</label>
  <textarea id="message" name="message" rows="4"></textarea>

  <button type="submit">Envoyer</button>
</form>
```

```css
form {
  display: flex;
  flex-direction: column;
  gap: 20px;
  padding: 20px;
  border: 1px solid green; /* Juste pour visualiser le conteneur */
}

label {
  font-weight: bold;
}

input,
textarea {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}

button {
  padding: 10px;
  border: 1px solid red; /* Juste pour visualiser les éléments */
}
```

Résultat: `flex-direction: column; gap: 20px;`

Nom :

E-mail :

Message :

Envoyer

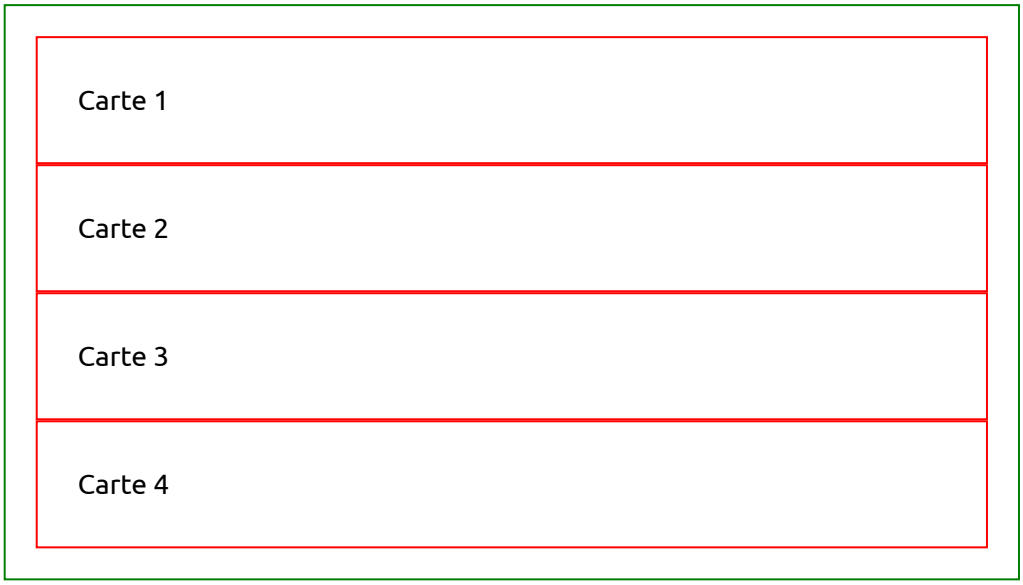
Grille de cartes responsive

```
<div class="card-grid">
  <div class="card">Carte 1</div>
  <div class="card">Carte 2</div>
  <div class="card">Carte 3</div>
  <div class="card">Carte 4</div>
</div>
```

```
.card-grid {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
}

.card {
  flex: 1 1 calc(25% - 20px); /* 25% moins le gap de 20px */
  border: 1px solid black;
  padding: 20px;
}

@media (max-width: 768px) {
  .card {
    flex: 1 1 calc(50% - 20px); /* Sur les écrans plus petits, utilise 2
cartes par ligne */
  }
}

@media (max-width: 480px) {
  .card {
    flex: 1 1 100%; /* Sur les écrans encore plus petits, une carte par
ligne */
```

```
    }
}
```

Résultat: `flex-wrap: wrap; gap: 20px;`



Avec ces exemples, vous pouvez voir comment Flexbox offre une grande flexibilité et facilite la création de diverses mises en page.

Avantages et inconvénients

Avantages

- **Simplicité:** Flexbox rend la mise en page, l'alignement et la distribution des espaces entre les éléments bien plus simples, notamment dans les situations où les tailles des éléments sont inconnues ou dynamiques.
- **Unidirectionnel:** Contrairement à CSS Grid, qui fonctionne sur deux axes (colonnes et lignes), Flexbox est axé principalement sur un seul axe à la fois, ce qui le rend plus intuitif dans certaines situations.
- **Adaptabilité :** Flexbox s'adapte bien aux différentes tailles d'écran, ce qui en fait une excellente option pour les designs responsives.
- **Alignement:** Flexbox offre des solutions simples pour aligner des éléments verticalement, une tâche qui était plutôt compliquée avec les techniques de mise en page plus anciennes.
- **Ordre des éléments:** Avec la propriété order, Flexbox permet de changer l'ordre visuel des éléments sans toucher à l'ordre du DOM, ce qui est pratique pour la responsivité.

Inconvénients :

- **Navigateurs :** Même si la prise en charge de Flexbox est désormais solide sur tous les navigateurs modernes, certains navigateurs plus anciens ne supportent pas toutes les fonctionnalités de Flexbox ou nécessitent des préfixes de fournisseur.

- **Complexité pour certains layouts:** Pour des mises en page plus complexes, comme les grilles à deux dimensions, CSS Grid est plus approprié.

Quand utiliser Flexbox vs d'autres méthodes (comme Grid ou Floats)

Flexbox:

- Lorsque vous travaillez avec des layouts linéaires (dans une seule direction).
- Pour aligner des éléments dans des barres de navigation, des en-têtes ou des pieds de page.
- Pour des composants d'interface utilisateur comme des listes, des groupes de boutons, etc.

CSS Grid:

- Pour des mises en page à deux dimensions (avec des colonnes et des lignes).
- Lorsque vous construisez des grilles complexes, comme des mises en page de magazine ou de portfolio.

Floats:

- Même si floats ont été largement remplacés par Flexbox et Grid pour la mise en page, ils peuvent encore être utiles pour des cas spécifiques comme faire flotter une image à l'intérieur d'un paragraphe de texte.

Limitations de Flexbox

1. **Unidirectionnel:** Bien que cela puisse être vu comme un avantage dans certains cas, cela limite aussi Flexbox en termes de capacité à gérer des layouts complexes à deux dimensions.

2. **Gestion des espaces:** Dans certaines situations, il peut être un peu complexe de comprendre comment Flexbox répartit l'espace lorsque flex-grow et flex-shrink sont utilisés.

3. **Limitation avec l'ancien contenu:** Intégrer Flexbox dans un site existant avec de l'ancien CSS (comme des floats) peut entraîner des comportements inattendus.

En conclusion, Flexbox est un outil puissant pour la mise en page web et est parfait pour des designs linéaires. Cependant, pour des grilles plus complexes, CSS Grid est souvent une meilleure option. Il est aussi essentiel de toujours considérer les besoins du projet et les navigateurs cibles avant de choisir une méthode de mise en page.

Ressources et outils

Outils de visualisation Flexbox en ligne

- Flexbox Playground
- Flexbox Froggy
- Flexbox Defense
- Flexbox Zombies
- Flexbox Cheatsheet
- Flexbox Patterns
- Flexbox Defense

- Flexbox Playground

Astuces et modèles Flexbox courants

- Flexbox Patterns
- Flexbox Cheatsheet
- Flexbox Grid
- Flexbox Grid Generator

Bibliothèques et frameworks utilisant Flexbox

- Bootstrap
- Tailwind CSS
- Bulma
- Materialize
- Foundation
- Semantic UI
- Pure CSS
- Skeleton
- Milligram