La Flexbox (boîte flexible)

La flexbox est un modèle de disposition très puissant qui va nous permettre de contrôler facilement et avec précision l'alignement, la direction, l'ordre et la taille de nos éléments (ou plus précisément de nos boites).

La mise en page avec les flexbox est basée sur les "directions de flux flexibles".

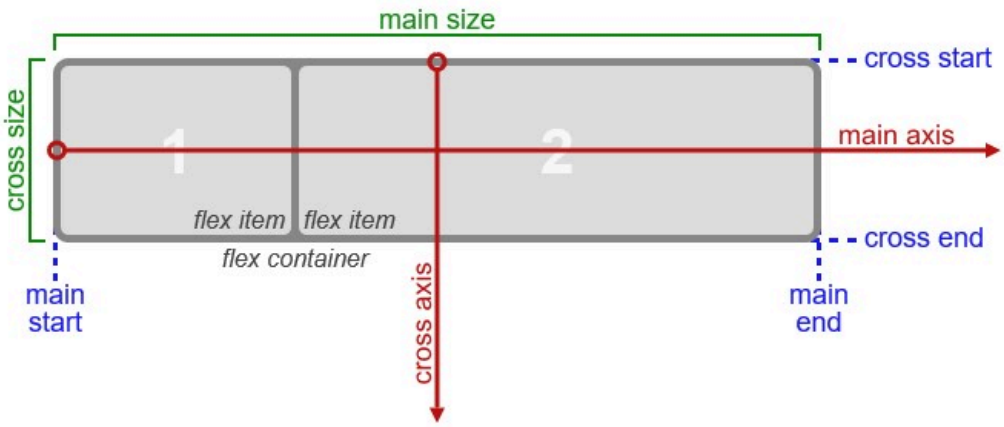
Le modèle des boites flexibles fait la distinction entre deux types de boites auxquelles on va pouvoir appliquer différentes propriétés :

- des conteneurs flexibles d'un côté
- des éléments flexibles ou « flex items » de l'autre.

Nous allons définir un conteneur flexible en attribuant un `display : flex` à un élément. Tous les éléments directement contenus dans ce conteneur (c'est-à-dire tous les enfants directs) vont alors automatiquement devenir des éléments flexibles (flex-items).

En plus de cette notion de conteneur contenant des enfants,
une deuxième notion est **très importante** celle des axes :
les flèches rouges de l'image main-axis et cross-axis

- principal => main-axis
- secondaire => cross-axis



Attention l'axe principal change en fonction de la valeur de la propriété :

flex-direction : row | row-reverse | column | column-reverse; [default row]

Par défaut, l'axe principal est l'axe horizontal (flex-direction : row) (et la direction va être celle de l'écriture) et l'axe secondaire est l'axe vertical (et la direction va être de haut en bas).

Si la direction est flex-direction : column alors l'axe principal est vertical et l'axe secondaire est horizontal

Ex: codepen Flexbox Alignement / axes
https://caniuse.com/

https://codepen.io/web-god/pen/vYJXJWQ

L’alignement / axe principal

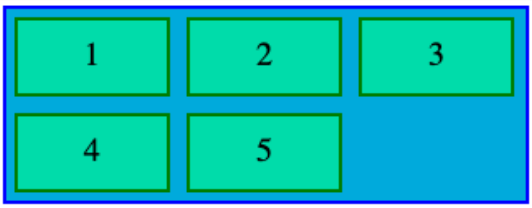
La propriété `justify-content` nous permet d’aligner les éléments d’un conteneur flexibles selon l’**axe principal**.

Les valeurs de justify-content :

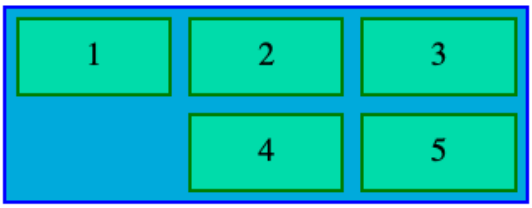
- `flex-start` : valeur par défaut. Les éléments vont être concentrés au début du conteneur (selon leur axe principal) ;
- `flex-end` : les éléments vont être concentrés à la fin du conteneur (selon leur axe principal) ;
- `center` : les éléments vont être centrés dans le conteneur (selon leur axe principal) ;
- `space-between` : les éléments vont être régulièrement distribués dans le conteneur. Les éléments se trouvant contre un bord du conteneur vont être collés au bord ;
- `space-around` : les éléments vont être régulièrement distribués dans le conteneur. Chaque élément va posséder le même espace et les espaces vont être cumulatifs entre deux éléments, ce qui fait que la taille de l’espace entre le conteneur et un élément contre le bord du conteneur sera deux fois plus petite qu’entre deux éléments ;
- `space-evenly` : les éléments vont être régulièrement distribués dans le conteneur. L’espace entre le bord du conteneur et un élément sera le même qu’entre deux éléments.

https://codepen.io/web-god/pen/wvqzQOQ

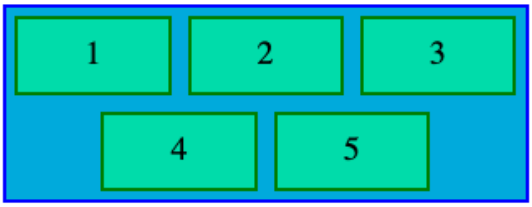
flex-start =>



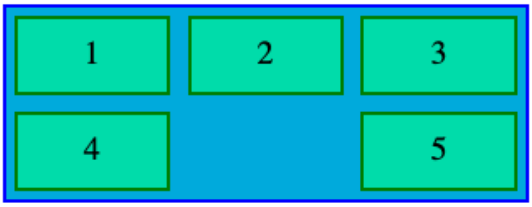
flex-end =>



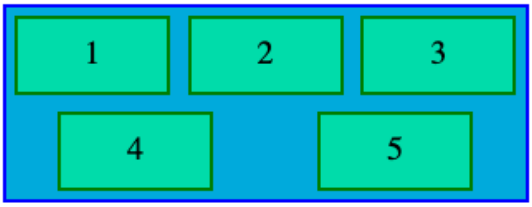
center =>



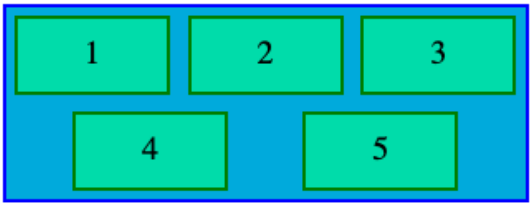
space-between =>



space-around =>



space-evenly =>



L’alignement / axe secondaire

L’alignement des éléments flexibles selon l’axe secondaire va pouvoir se faire de trois façons et avec plusieurs propriétés :

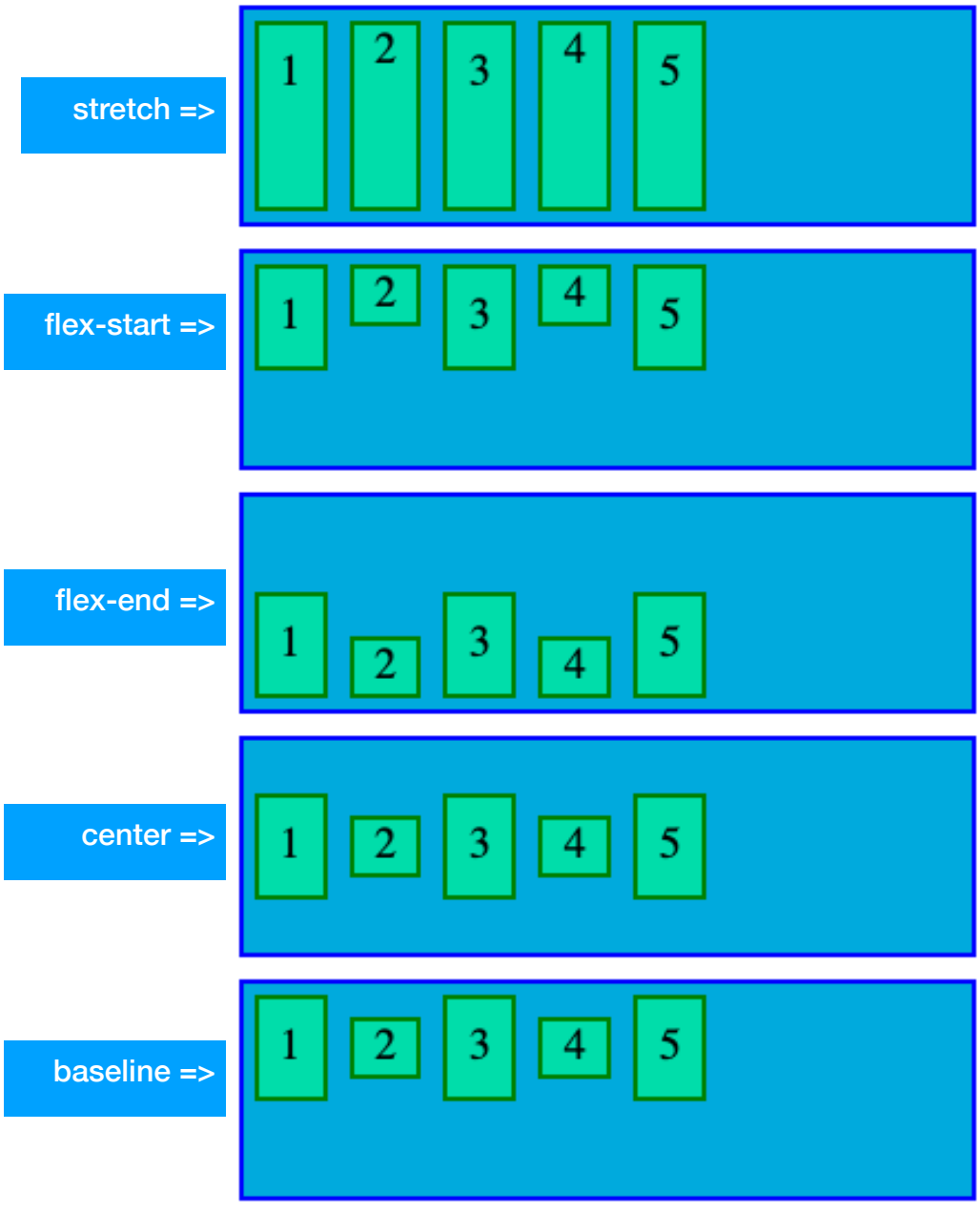
- align-items : alignement des éléments au sein d’une ligne (ou d’une colonne selon les axes principal et secondaires choisis) dans l’axe secondaire ;
  Equivalent de `justify-content` pour l’axe principal

- align-item : alignement des lignes (ou des colonnes selon l’axe) les unes par rapport aux autres dans l’axe secondaire et ne va donc avoir un effet que si nous avons plusieurs lignes (ou colonnes)

- align-self : alignement d’un élément flexible en particulier dans l’axe secondaire.

align-items

- `stretch` : valeur par défaut. Les éléments vont s'étirer dans leur axe secondaire jusqu'à remplir tout l'espace disponible ;
- `flex-start` : les éléments vont être placés au début de leur conteneur en fonction de l'axe secondaire ;
- `flex-end` : les éléments vont être placés à la fin de leur conteneur en fonction de l'axe secondaire ;
- `center` : les éléments vont être placés au milieu de leur conteneur en fonction de l'axe secondaire ;
- `baseline` : les éléments vont être alignés dans leur axe secondaire de telle sorte que leurs lignes de base (ligne imaginaire sur laquelle est écrit le texte) soient alignées.

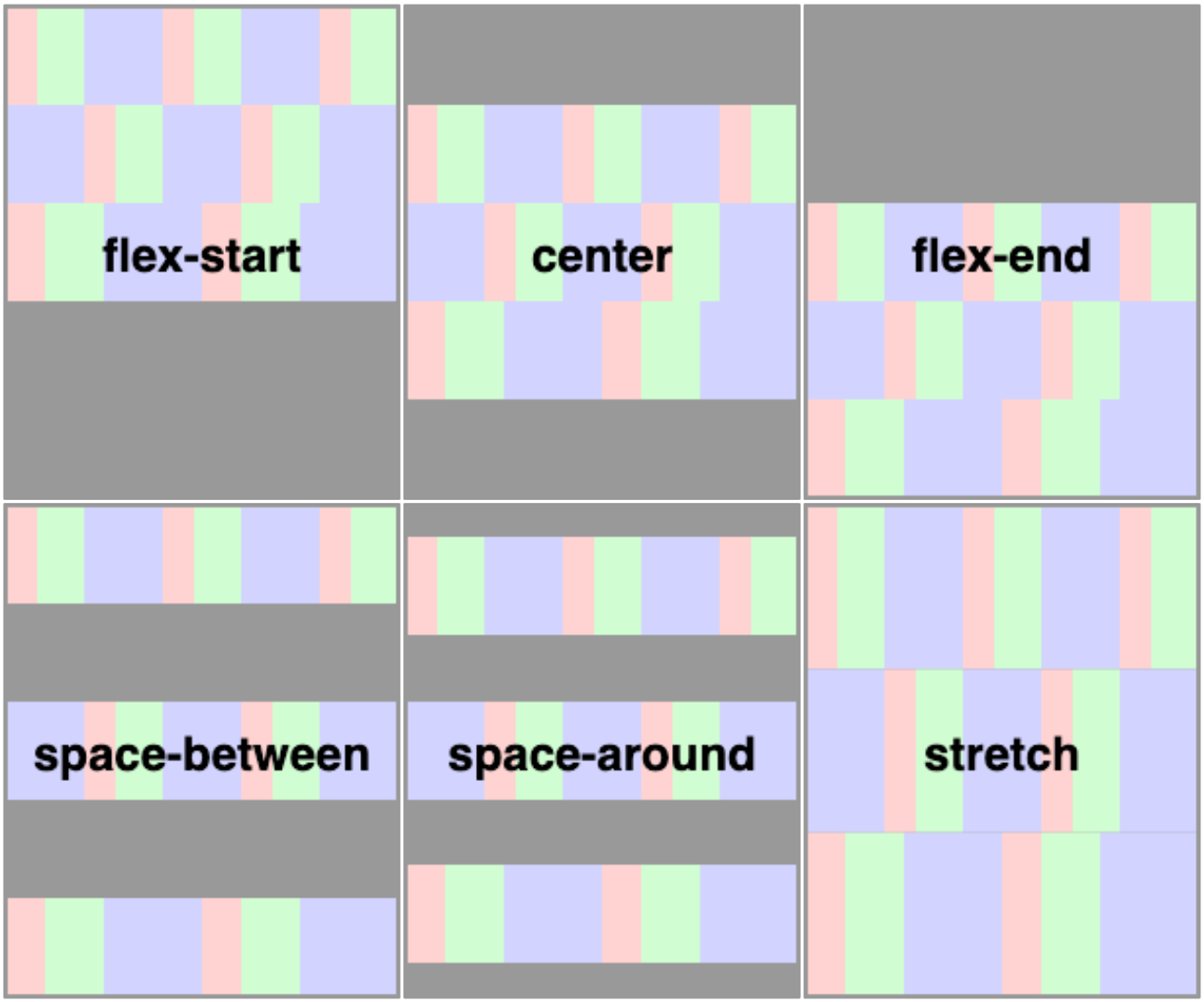


align-content

Ceci peut être déroutant, mais align-content détermine l'espace entre les lignes, alors que align-items détermine comment les éléments dans leur ensemble sont alignées à l'intérieur du conteneur.

Quand il n'y a qu'une seule ligne, align-content n'a aucun effet.

- flex-start : Les lignes sont amassées dans le haut du conteneur.
- flex-end: Les lignes sont amassées dans le bas du conteneur.
- center : Les lignes sont amassées dans le centre vertical du conteneur.
- space-between : Les lignes s'affichent avec un espace égal entre eux.
- space-around : Les lignes s'affichent avec un espace égal autour d'eux.
- stretch : Les lignes sont étirées pour s'adapter au conteneur.



La propriété flex

Il est important de se rappeler que :

> le **contenu** d'un élément flex a un impact sur la façon dont flex-grow, flex-shrink et flex-basis fonctionnent ensemble.

Par défaut cette propriété est fixée à :

```
flex: 0 1 auto; grow | shrink | basis
```

C'est l'équivalent de :

```
.child {
  flex-grow: 0;
  flex-shrink: 1;
  flex-basis: auto;
}
```

On pourrait aussi l'imaginer ainsi

```
.child {
  flex: [max] [min] [ideal size];
}
```

flex-grow

La valeur de flex-grow est fixée à **0** car nous voulons que l'élément s'agrandisse en fonction du contenu qu'il contient.

Si nous changeons la valeur de flex-grow à **1** tous les éléments grandiront pour occuper une partie égale de l'élément parent.

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

flex-shrink et flex-basis gardent leurs valeurs par défaut.

flex-shrink

flex-shrink indique au navigateur quelle doit être la taille minimale d'un élément.
La valeur par défaut est 1, ce qui revient à dire "occupe la même quantité d'espace à tout moment".

Si nous changeons la valeur de flex-shrink à **0** nous disons à cet élément de ne pas rétrécir du tout maintenant.

```
.child {
  flex-shrink: 0;
}
```

flex-basis

flex-basis est la façon dont nous indiquons à un élément de s'en tenir à une taille idéale. Par défaut, elle est définie sur **auto**, ce qui signifie "Utilisez ma hauteur ou ma largeur".

Auto veut dire que la taille idéale de notre élément est définie par son contenu.

Pour que tous les éléments occupent tout l'espace du parent, nous pouvons définir les éléments enfants sur :

width : 100%

ou

flex-basis : 100%

ou

flex-grow : 1

Chacune de ces valeurs abrégées a un impact sur les autres et c'est pourquoi il est recommandé d'écrire flex-basis en premier lieu.

Si nous changeons la valeur de flex-basis à **1000px** nous disons à cet élément de ne pas du tout rétrécir maintenant.

```
.child {
  flex-basis: 1000px;
}
```

Nous disons au navigateur de définir flex-basis à 1000px ou "s'il vous plaît, s'il vous plaît, essayez d'occuper 1000px d'espace".

Si ce n'est pas possible, l'élément occupera cet espace proportionnellement aux autres éléments.

Parce que ces 3 propriétés sont liées, il est recommander d'utiliser le raccourci flex

Ex : flex : 2 => l'élément occupera 2 fois plus d'espace que les autres éléments

Pour résumé :

A. Essayez d'utiliser l'abréviation flex

B. N'oubliez pas les valeurs max, min et taille idéale.

C. N'oubliez pas que le contenu d'un élément peut également avoir un impact sur la façon dont ces valeurs fonctionnent ensemble.

Voir codepen Flexbox flex grow | shrink | basis

https://flexboxfroggy.com/#fr
https://css-tricks.com/snippets/css/a-guide-to-flexbox/
https://the-echoplex.net/flexyboxes/