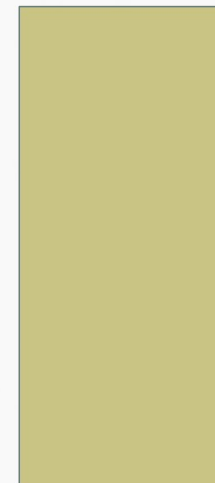


FLEXBOX

[HTTPS://DEVELOPER.MOZILLA.ORG/FR/DOCS/WEB/CSS/CSS_FLEX
IBLE_BOX_LAYOUT/BASIC_CONCEPTS_OF_FLEXBOX](https://developer.mozilla.org/fr/docs/web/css/css_flexible_box_layout/basic_concepts_of_flexbox)



QU'EST CE QU'UNE FLEXBOX ?

- Une Flexbox c'est une boîte conteneur (**flex-container**) et qui contient d'autres boîtes(**flex-items**).
- À l'aide des propriétés du flex-container, on va contrôler : la répartition des flex-items dans le flex-container, leur taille, leur position, leur étirement, leur rétrécissement, leur ordre, le retour à la ligne.
- **Le contrôle se limite aux flex-items pas à leur contenu.**

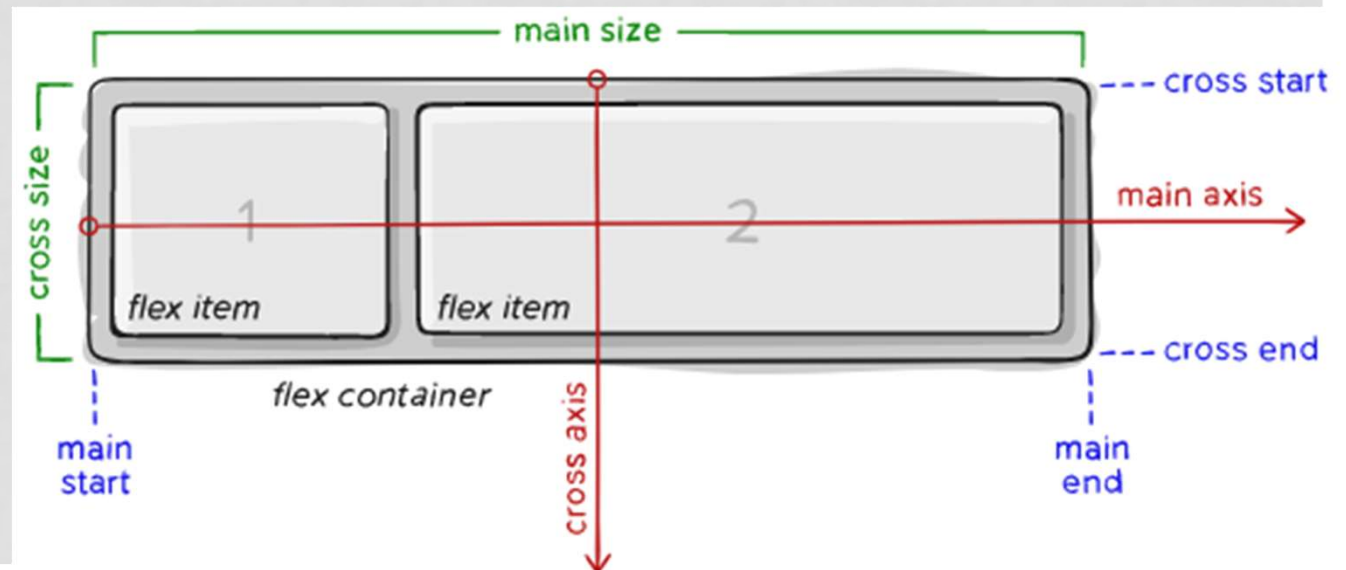
FLEX-CONTAINER / FLEX-ITEMS

- Pour créer un **flex-container**, on donne à la propriété **display** la valeur **flex**.
- les enfants directs d'un **flex-container** deviennent des **flex-items**.

```
.container  
{  
|   display: flex;  
}
```

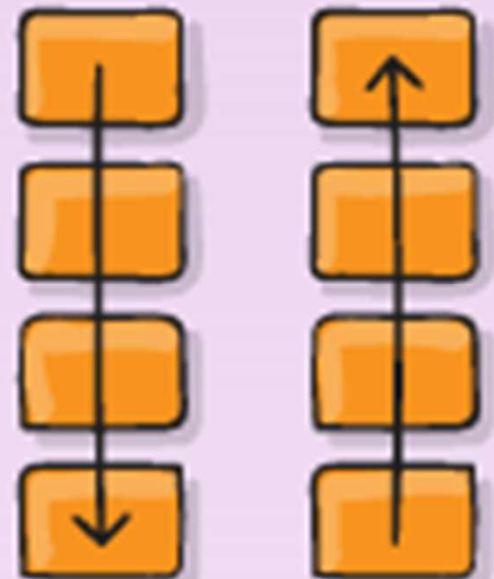
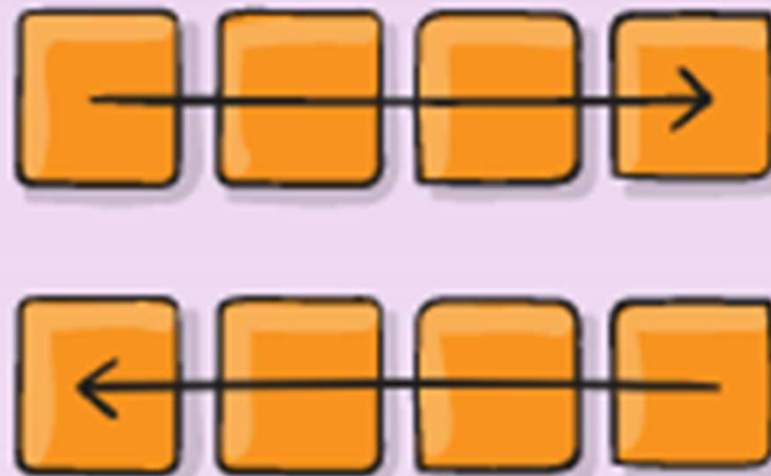
LES DEUX AXES

- La répartition des flex-items se fait selon deux axes :
 - L'axe principal (*main axis*) est horizontal et la répartition des flex-items se fait de la gauche vers la droite.
 - L'axe secondaire (*cross axis*) est perpendiculaire à l'axe principal et la répartition se fait du haut vers le bas.
- Par défaut, la répartition des flex-items suit le sens d'écriture (français, arabe...)



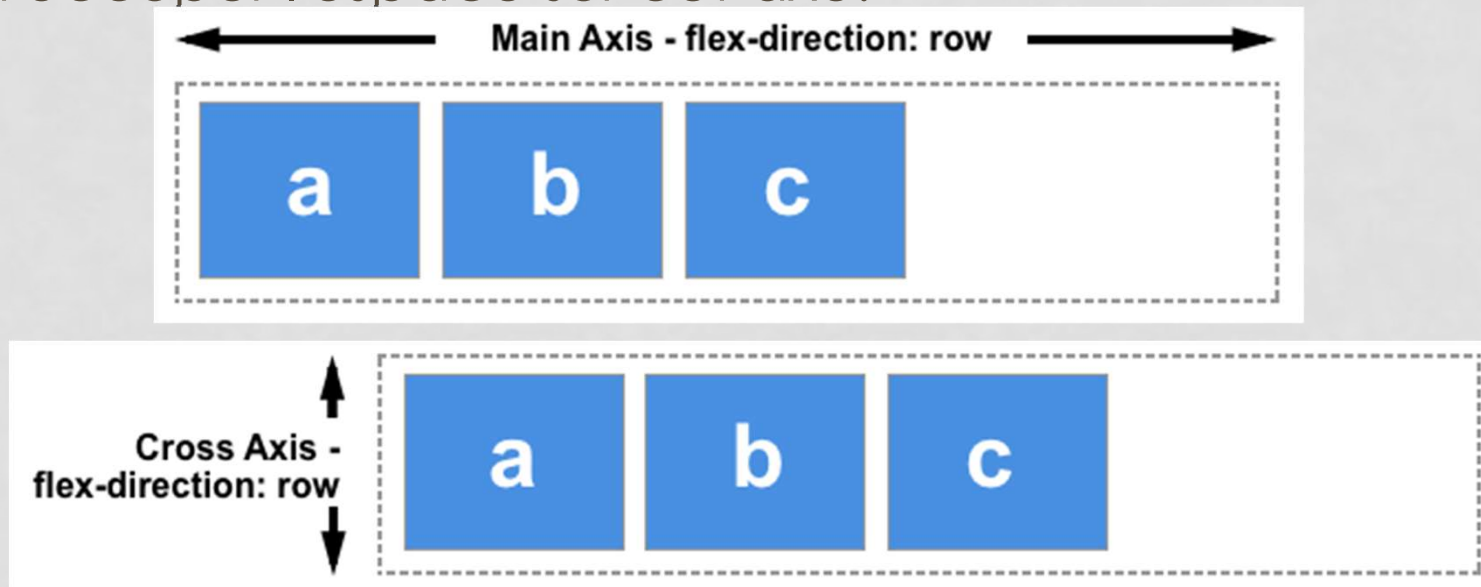
FLEX-DIRECTION

- Cette propriété permet la répartition des flex-items dans le flex-container selon sa valeur :
 - flex-direction: row;
 - flex-direction: row-reverse;
 - flex-direction: column;
 - flex-direction: column-reverse;



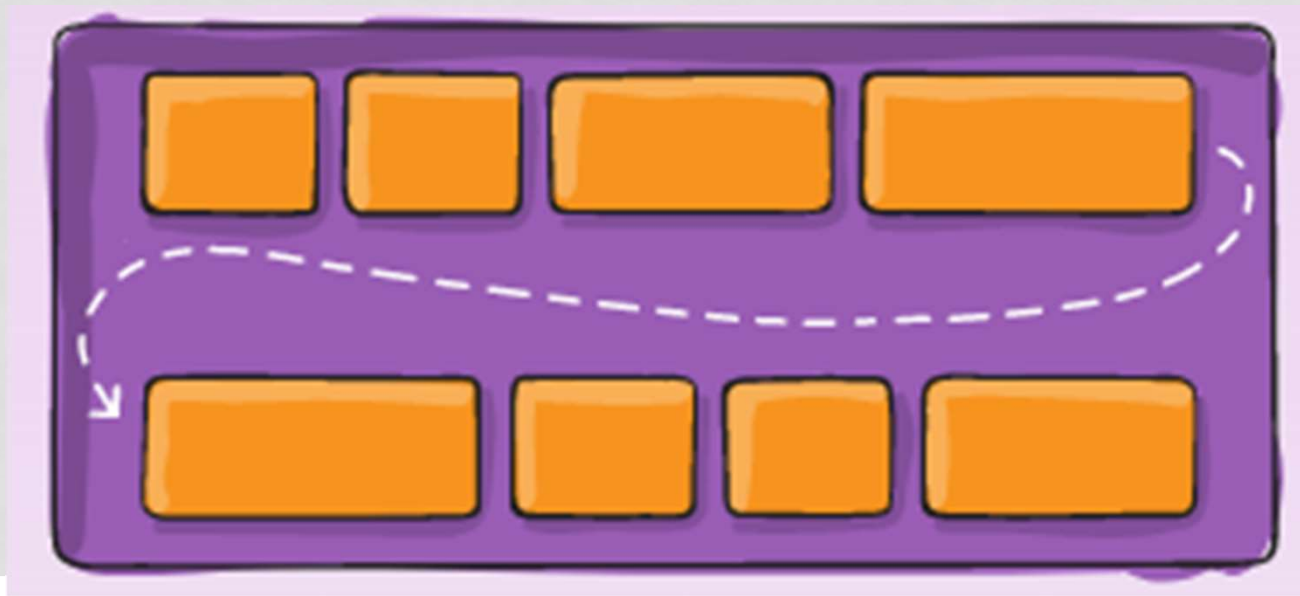
FLEX-DIRECTION

- flex-direction: 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.



FLEX-WRAP

- Flex-wrap: nowrap (par défaut) : tous les éléments flexibles seront sur une seule ligne
- Flex-wrap: wrap: les éléments flexibles s'enrouleront sur plusieurs lignes, de haut en bas.
- Flex-wrap: wrap-reverse: les éléments flexibles s'enrouleront sur plusieurs lignes de bas en haut.

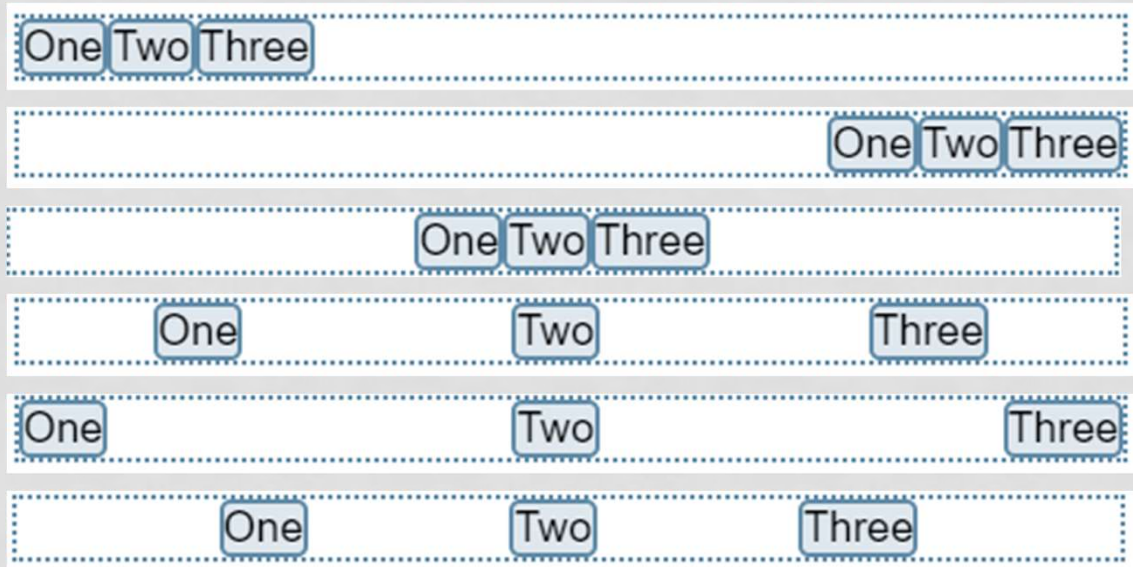


FLEX-FLOW

- La propriété flex-flow est le raccourci des propriétés flex-direction et flex-wrap,
 - Exemple : flex-flow: row wrap;
- La première valeur sera utilisée pour flex-direction et la seconde pour flex-wrap.

JUSTIFY-CONTENT

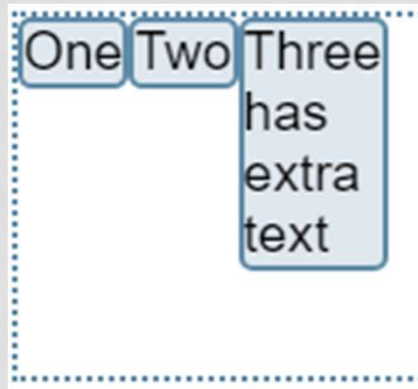
- justify-content est utilisée afin d'aligner les flex-items le long de l'axe principal dans la direction définie par flex-direction. Cette propriété a 5 valeurs :
- flex-start
- flex-end
- center
- space-around
- space-between
- space-evenly



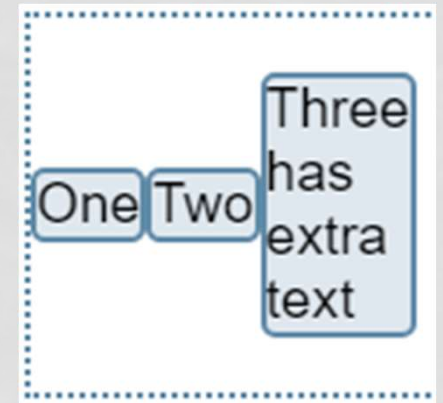
ALIGN-ITEMS

- align-items permet d'aligner les flex-items le long de l'axe secondaire.

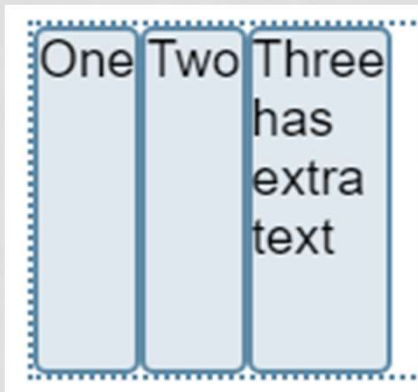
```
.box {  
  display: flex;  
  align-items: flex-start;  
}
```



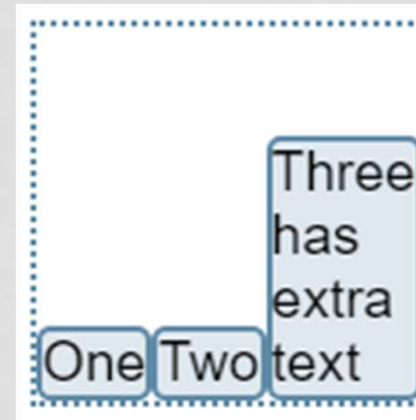
```
.box {  
  display: flex;  
  align-items: center;  
}
```



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

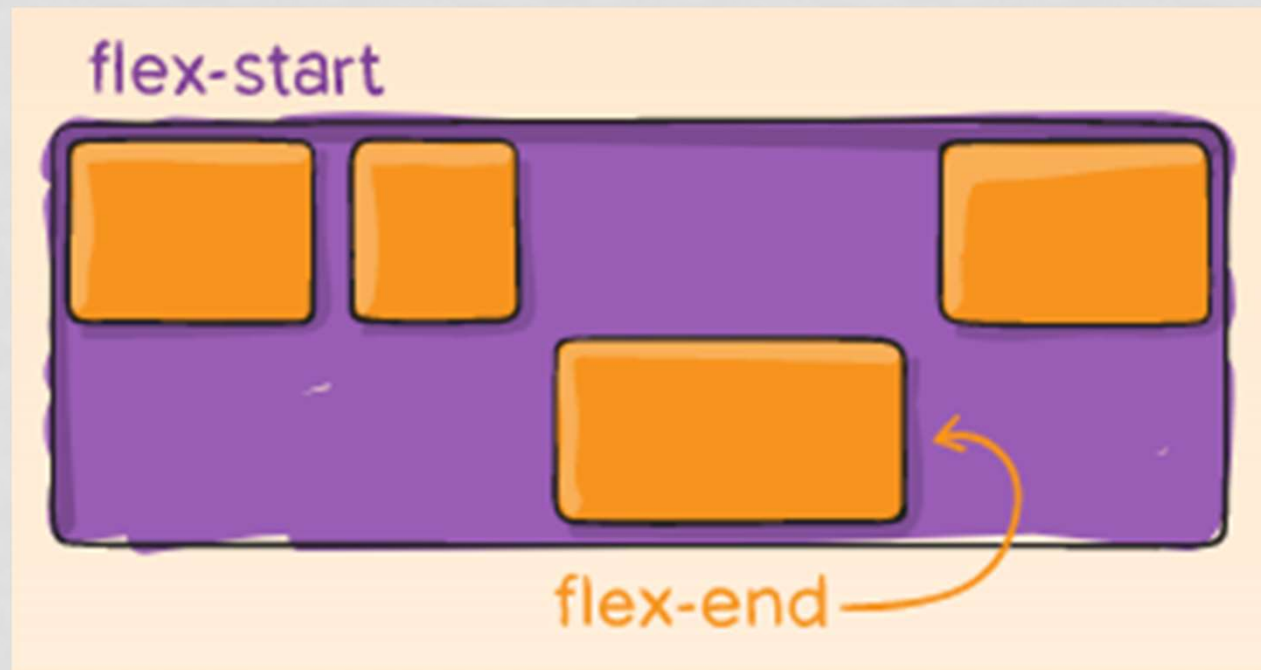


```
.box {  
  display: flex;  
  align-items: flex-end;  
}
```



ALIGN-SELF

- remplace l'alignement par défaut (ou celui spécifié par align-items) pour le flex-item individuel.
Exemple : align-self : flex-end;



ORDER

- Par défaut, les flex-items sont disposés dans l'ordre source. Cependant, la propriété `order` contrôle l'ordre dans lequel ils apparaissent dans le flex-container.

- Exemple:



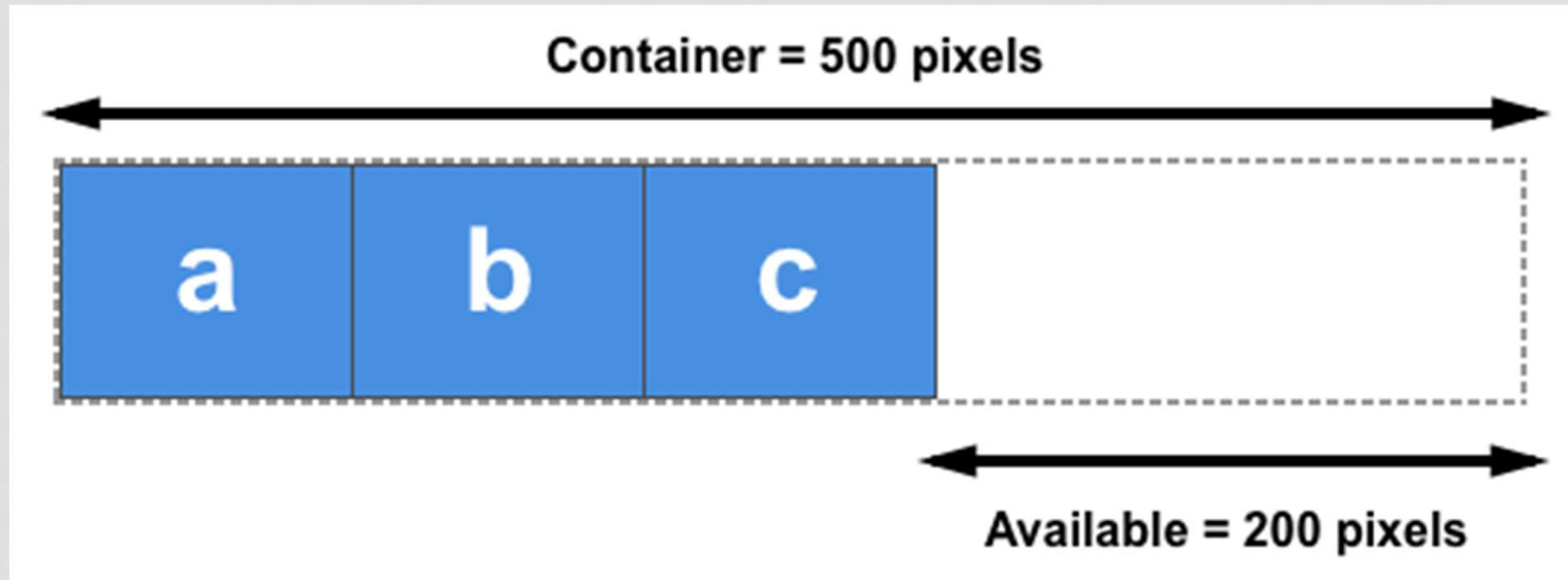
```
<div class="box">
  <div class="one">One</div>
  <div class="two">Two</div>
  <div class="three">Three</div>
</div>
```

```
.one {
  order: 2;
}

.two {
  order: 3;
}

.three {
  order: 1;
}
```

GÉRER L'ESPACE DISPONIBLE



- flex-basis
- flex-grow
- flex-shrink

FLEX-BASIS

- Flex-basis: auto;
- Cela définit la taille par défaut d'un flex-item avant que l'espace restant ne soit distribué.
- un flex-item prend sa taille si elle est définie explicitement.
- Sinon, il prend la taille de son contenu.
- Exemple :



FLEX-GROW

- La propriété flex-grow permet à un flex-items de s'étendre à partir de la mesure de flex-basis.
- L'élément sera étiré et occupera l'espace disponible sur cet axe ou une part de cet espace si les autres flex-items peuvent s'étendre également.
- Exemple :



```
.one { flex-grow: 1; }    .two { flex-grow: 2; }    .three { flex-grow: 3; }
```


FLEX-SHRINK

- S'il n'y a pas suffisamment d'espace dans le flex-container pour disposer les éléments, le flex-shrink contrôle la façon dont l'espace sera réduit.

FLEX

- La propriété flex est le raccourci des propriétés flex-grow, flex-shrink et flex-basis,.
 - Exemple : flex: 1 1 auto;
- La première valeur sera utilisée pour flex-grow, la seconde pour flex-shrink et la troisième pour flex-basis.
- Exemple:

One

Two

Three

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

```
.box > * {  
  width: 200px;  
}
```

```
.one {  
  flex: 1 1 auto;  
}
```

```
.two {  
  flex: 1 0 auto;  
}
```

```
.three {  
  flex: 1 0 auto;  
}
```

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

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

```
.box > * {  
  width: 200px;  
}
```

```
.one {  
  flex: 1 0 auto;  
}
```

```
.two {  
  flex: 1 0 auto;  
}
```

```
.three {  
  flex: 1 0 auto;  
}
```

One

Two

Three

MÉDIA QUERIES



DÉFINITION

- Le Média Queries est une technique qui vise à changer le design d'un site Internet selon les caractéristiques de l'écran utilisé :
 - Width (largeur de la fenêtre d'affichage),
 - Height (hauteur de la fenêtre d'affichage),
 - Device-height (hauteur de l'écran),
 - Device-width (largeur de l'écran),
 - Orientation (orientation de la zone d'affichage : portrait ou paysage),
- C'est un élément important du CSS3.

SON UTILISATION

- Dans la feuille HTML, on doit ajouter la balise :
- `<meta name="viewport" content="width=device-width, initial-scale=1.0">`
 - **viewport** donne au navigateur des instructions sur la façon de contrôler les dimensions et la mise à l'échelle de la page.
 - La **width=device-width** partie définit la largeur de la page pour qu'elle suive la largeur de l'écran de l'appareil (qui varie en fonction de l'appareil).
 - La **initial-scale=1.0** partie définit le niveau de zoom initial lorsque la page est chargée pour la première fois par le navigateur.



>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempore eum soluta nobis eleifend congue nihil imperdiet domine



>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempore eum soluta nobis eleifend congue nihil imperdiet domine

Sans la balise meta viewport

Avec la balise meta viewport

SON UTILISATION

- Dans une feuille CSS :

@media **not** | **only** *mediatype* and
(*mediafeature*) and | or | not (*mediafeature*)

```
{  
  CSS-Code;  
}
```

Avec

mediatype	<i>mediafeature</i>
all, screen, print	max-width, min-width, max-height, min-height, Orientation (portrait landscape)

SON UTILISATION

- Exemple :

```
@media screen and (max-width: 800px)
{
  body { background-color: green; }
}
```

```
@media screen and (min-width: 801px)
{
  body { background-color: red; }
}
```

- Dans plusieurs feuilles CSS :

Exemple:

```
<link rel="stylesheet" media="screen and (min-width: 901px)" href="widescreen.css">
```

```
<link rel="stylesheet" media="screen and (max-width: 900px)" href="smallscreen.css">
```

RESPONSIVE DESIGN





Vidéo

ANIMATION



NOTIONS DE BASE

- Définition :
 - Animation est le fait de changer la valeur d'une propriété CSS par une autre.
- Types d'animations :
 - Transition : une propriété qui change lors d'un événement (généralement, le survol par la souris)
 - Transformation : modifie l'apparence des éléments: rotation, échelle, translation...
 - @keyFrames : cette règle donne le contrôle sur le déclenchement et la progression du changement de valeur des propriétés animées. L'animation se déclenche pendant le chargement de la page.

TRANSITION

- Example:

```
<div class="mag">
  
  
  
  
  
  
</div>
```

```
.mag img
{ width:100px;
}


.mag1:hover
{ width:300px;
}
```


LES PROPRIÉTÉS DE LA TRANSITION

- transition-property : le nom de la propriété qu'on veut changer;
- transition-duration : la durée de la transition en seconde.
- transition-delay : démarre la transition après un certain temps en seconde
- transition-timing-function ; définis la vitesse de transition. Elle 5 valeurs:
 - ease : valeur par défaut. transition lente au début puis s'accélère au milieu et se termine lentement ;
 - linear : transition va à la même vitesse du début à la fin ;
 - ease-in : transition avec un départ lent puis s'accélère ensuite ;
 - ease-out : transition qui va ralentir à la fin ;
 - ease-in-out : transition démarre plus lentement au début puis s'accélère au milieu et se termine lentement

EXAMPLE

```
<div class="mag">
  
  
  
  
  
  
</div>
```

```
.mag img
{
  width:100px;
  height:100px;
  border:  red solid 2px;
}
```

```
.mag1:hover
{
  width:300px;
  height:300px;
  border:  green solid 5px;
  transition-property: height;
  transition-duration: 4s;
  transition-delay: 2s;
  transition-timing-function: linear;
}
```

TRANSFORMATION

- La propriété transform a 5 valeurs possibles :
 - Rotation : `transform: rotate(30deg)`
 - Échelle : `transform: scale(2, 2)`
 - Translation : `transform : translate(10px, -3em)`
 - Inclinaison : `transform: skewX(10deg)`, `transform: skewY(2.1rad)`


EXAMPLE



```
.mag2:hover  
{ transform: rotate(30deg);  
}  
.mag3:hover  
{ transform: scale(2, 2);  
}  
.mag4:hover  
{ transform : translate(50px, 50px);  
}  
.mag5:hover  
{ transform: skewX(50deg);  
}
```


LES PROPRIÉTÉS DE L'ANIMATION


- **animation-name** : donner un nom à l'animation
- **animation-duration** : la durée totale de l'animation en seconde,
- **animation-timing-function** : la vitesse de l'animation comme **transition-timing-function**
- **animation-delay** : retarder le départ de l'animation
- **animation-iteration-count** : indiquer le nombre d'exécutions de l'animation,
- **animation-direction** pour indiquer le sens d'exécution de l'animation : normal, alternate, reverse, alternate-reverse.





EXAMPLE 1

```
.mag6
{ border:  red solid 2px;
  animation-name: anima;
  animation-duration: 4s;
  animation-delay: 2s;
  animation-iteration-count: 3;
  animation-direction: alternate;
}

@keyframes anima {
  from {border:  red solid 2px;}
  to {border:  red solid 20px;}
}
```


EXAMPLE 2

```
.mag6
{ border:  red solid 2px;
  animation-name: anima;
  animation-duration: 4s;
  animation-delay: 2s;
  animation-iteration-count: 3;
  animation-direction: alternate;
}

@keyframes anima {
  0%    {border:  red solid 2px;}
  25%   {border:  green solid 10px;}
  50%   {border:  blue solid 15px;}
  100%  {border:  black solid 20px;;}
}
```

TEMPLATE

DÉFINITION

- un **Template** est un moyen de séparer le contenu rédactionnel (contenu textuel) de la forme (la manière dont il est présenté).
- Un **Template** est un modèle où **seuls certains éléments sont modifiables** (le contenu texte, les images, les couleurs et le fond).

POURQUOI UTILISER UN TEMPLATE?

- Utiliser un Template pour créer un nouveau site Internet a deux avantages majeurs:
 - Il permet de gagner beaucoup de temps
 - Il permet de réduire le prix du site Internet
- **C'est donc un choix idéal pour les nouvelles entreprises avec un budget limité.**

QU'ELLES SONT LES INCONVÉNIENTS?

- Le principal inconvénient c'est que ton site ne sera pas unique. La structure ressemblera beaucoup aux autres sites Internet ayant utilisé le même Template. Il y aura donc un effet de déjà vu. Toute fois, pour palier à ce défaut, tu pourras personnaliser le Template.

COMMENT PERSONNALISER UN TEMPLATE?

- Pour différencier un peu le Template utilisé, il faudra y apporter ta touche personnelle. Cela pourra être fait en travaillant sur plusieurs points:
 - Ajuster les couleurs du site aux couleurs de ton logo
 - Ajouter les sections propres à ton entreprise
 - Rédiger des textes uniques
 - Insérer de belles photos qui illustrent parfaitement ton contenu

À QUI EST DESTINÉ LES TEMPLATES?

- si le message est plus important que l'image
- si tu as un budget réduit
- si tu souhaites avoir un site Internet le plus rapidement possible
- si tu souhaites tester le marché avant d'investir davantage

EXAMPLE

- <https://www.free-css.com/free-css-templates/page245/life-care>