CSS Flex BOX

For

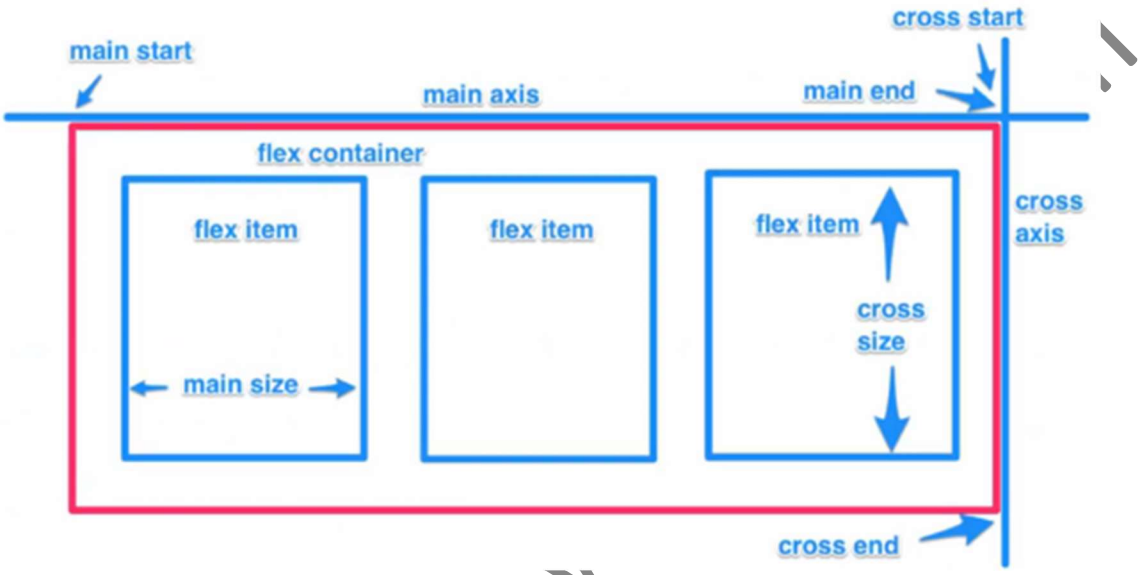
MUHAMMAD JUNAID

0304-1659294 | junaidfazal08@gmail.com

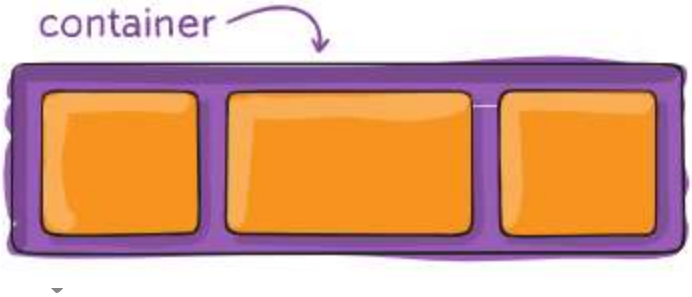
AL-JUNAID INSTITUTE OF GROUP

What is Flex Box?

- Aims at providing an efficient and easy way to align items in a Container
- It is Direction Agnostic.



Properties for the Parent (flex container)



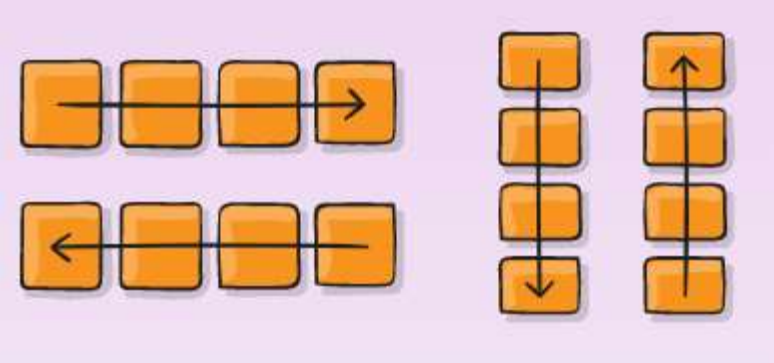
This defines a flex container; inline or block depending on the given value. It enables a flex context for all its direct children.

AL-JUNAID INSTITUTE OF GROUP

```
.container {
  display: flex; /* or inline-flex */
}
```

flex-direction:

This establishes the main-axis, thus defining the direction flex items are placed in the flex container. Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.



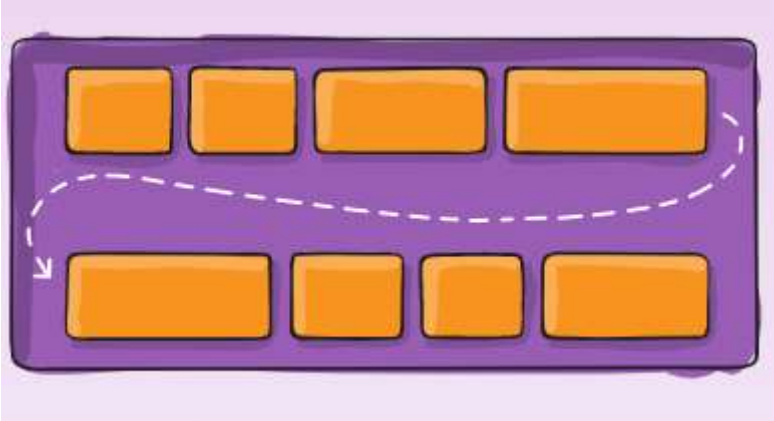
```
.container {
  flex-direction: row | row-reverse | column | column-reverse;
}
```

- row (default): left to right in ltr; right to left in rtl
- row-reverse: right to left in ltr; left to right in rtl
- column: same as row but UP and DOWN
- column-reverse: same as row-reverse but bottom to top

AL-JUNAID INSTITUTE OF GROUP

flex-wrap

By default, flex items will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property.



```
.container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

- nowrap (default): all flex items will be on one line
- wrap: flex items will wrap onto multiple lines, from top to bottom.
- wrap-reverse: flex items will wrap onto multiple lines from bottom to top.

HTML:

```
<ul class="flex-container nowrap">
  <li class="flex-item">1</li>
  <li class="flex-item">2</li>
  <li class="flex-item">3</li>
  <li class="flex-item">4</li>
  <li class="flex-item">5</li>
  <li class="flex-item">6</li>
  <li class="flex-item">7</li>
  <li class="flex-item">8</li>
</ul>
<ul class="flex-container wrap">
  <li class="flex-item">1</li>
```

For More: vulmshelp.com

```
  <li class="flex-item">2</li>
  <li class="flex-item">3</li>
  <li class="flex-item">4</li>
  <li class="flex-item">5</li>
  <li class="flex-item">6</li>
  <li class="flex-item">7</li>
  <li class="flex-item">8</li>
</ul>
<ul class="flex-container wrap-reverse">
  <li class="flex-item">1</li>
  <li class="flex-item">2</li>
  <li class="flex-item">3</li>
  <li class="flex-item">4</li>
  <li class="flex-item">5</li>
  <li class="flex-item">6</li>
  <li class="flex-item">7</li>
  <li class="flex-item">8</li>
</ul>
```

CSS:

```
.flex-container {
  padding: 0;
  margin: 0;
  list-style: none;
  border: 1px solid silver;
  -ms-box-orient: horizontal;
  display: -webkit-box;
  display: -moz-box;
  display: -ms-flexbox;
  display: -moz-flex;
  display: -webkit-flex;
  display: flex;
}
.nowrap  {
  -webkit-flex-wrap: nowrap;
  flex-wrap: nowrap;
}
.wrap    {
  -webkit-flex-wrap: wrap;
  flex-wrap: wrap;
```

For Moe Visit: vulmshelp.com

```
}
.wrap li {
  background: gold;
}

.wrap-reverse        {
  -webkit-flex-wrap: wrap-reverse;
  flex-wrap: wrap-reverse;
}
.wrap-reverse li {
  background: deepskyblue;
}

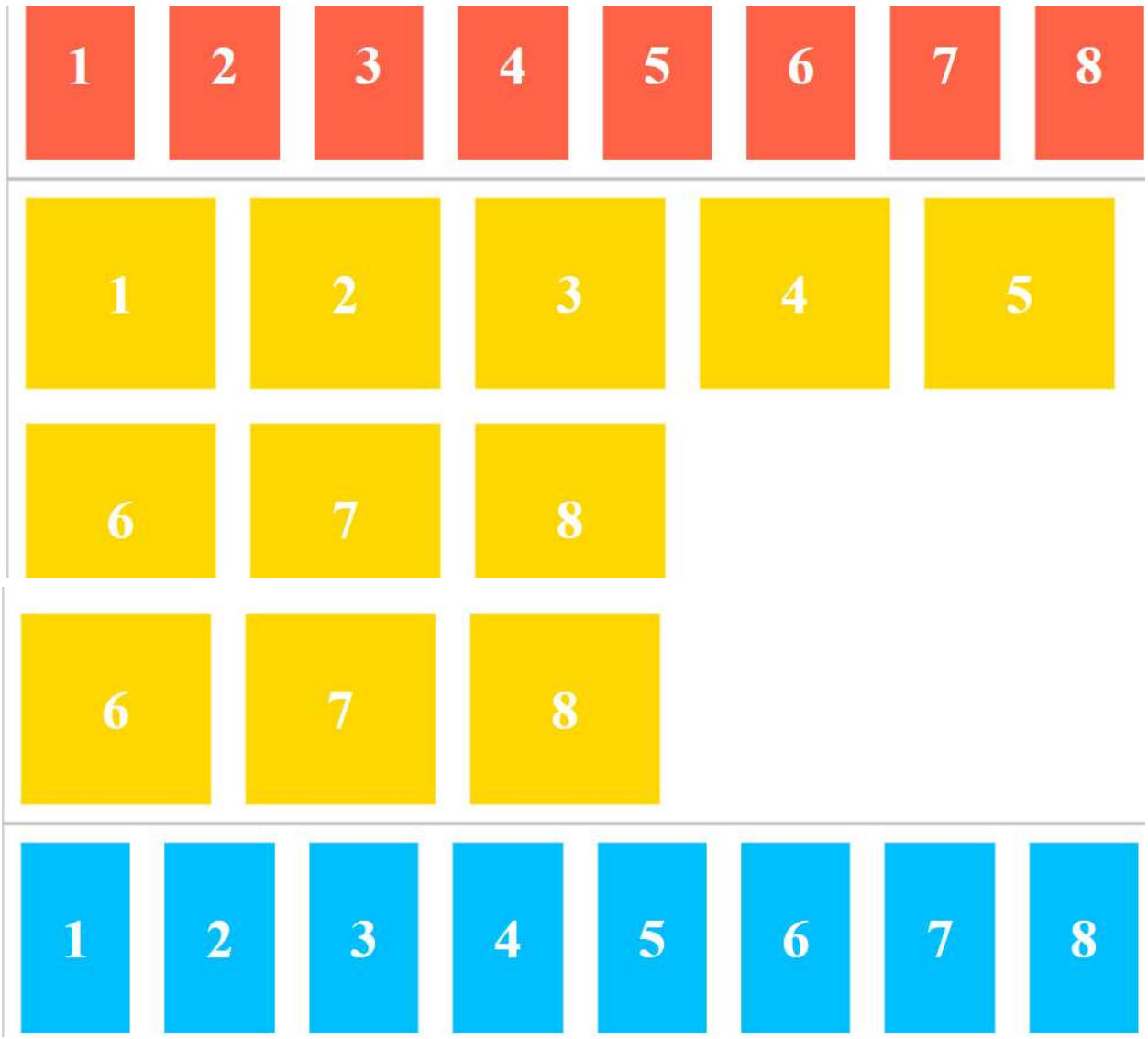
.flex-item {
  background: tomato;
  padding: 5px;
  width: 100px;
  height: 100px;
  margin: 10px;

  line-height: 100px;
  color: white;
  font-weight: bold;
  font-size: 2em;
  text-align: center;
}
```

OUTPUT:

For More Visit: vulmshelp.com

AL-JUNAID INSTITUTE OF GROUP



flex-flow

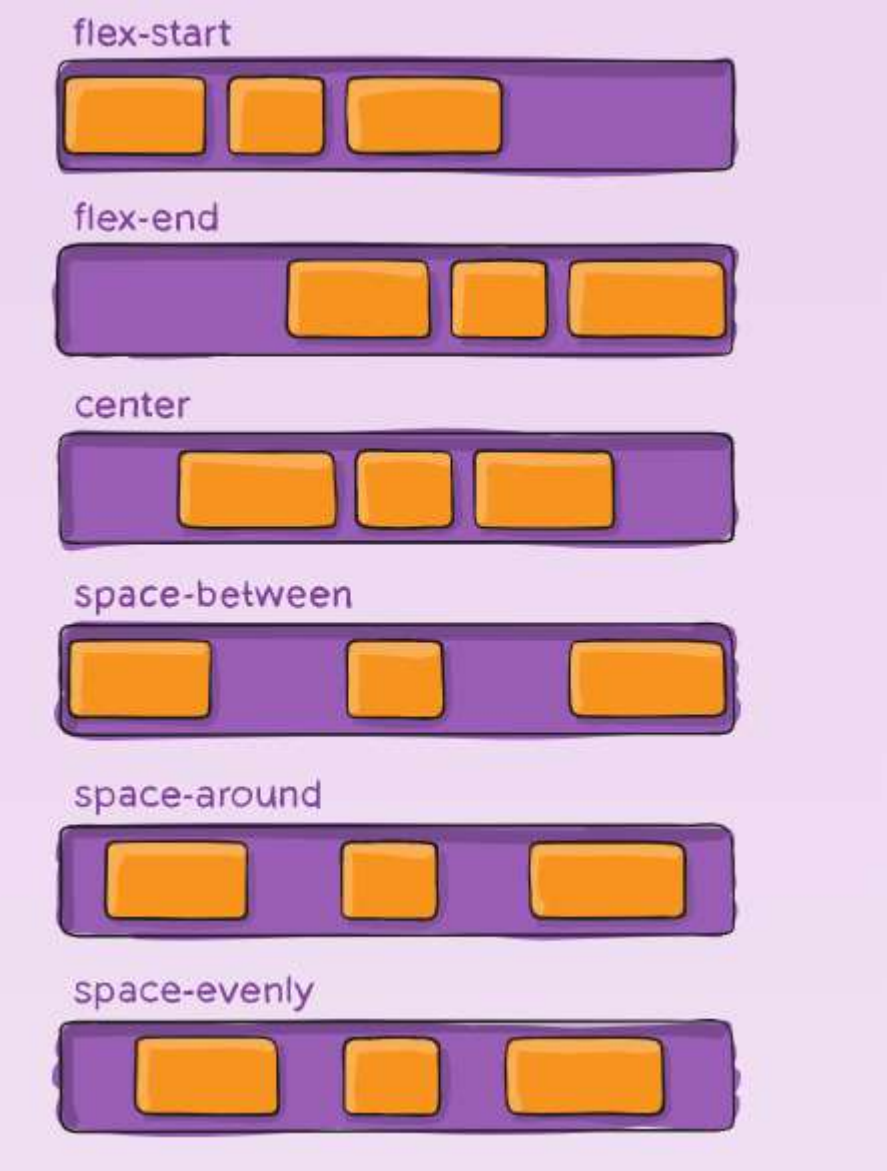
This is a shorthand for the flex-direction and flex-wrap properties, which together define the flex container's main and cross axes. The default value is row no wrap.

```
.container {
  flex-flow: column wrap;
}
```

AL-JUNAID INSTITUTE OF GROUP

justify-content

This defines the alignment along the main axis. It helps distribute extra free space leftover when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.




```
.container {
  justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right ... + safe | unsafe;
}
```

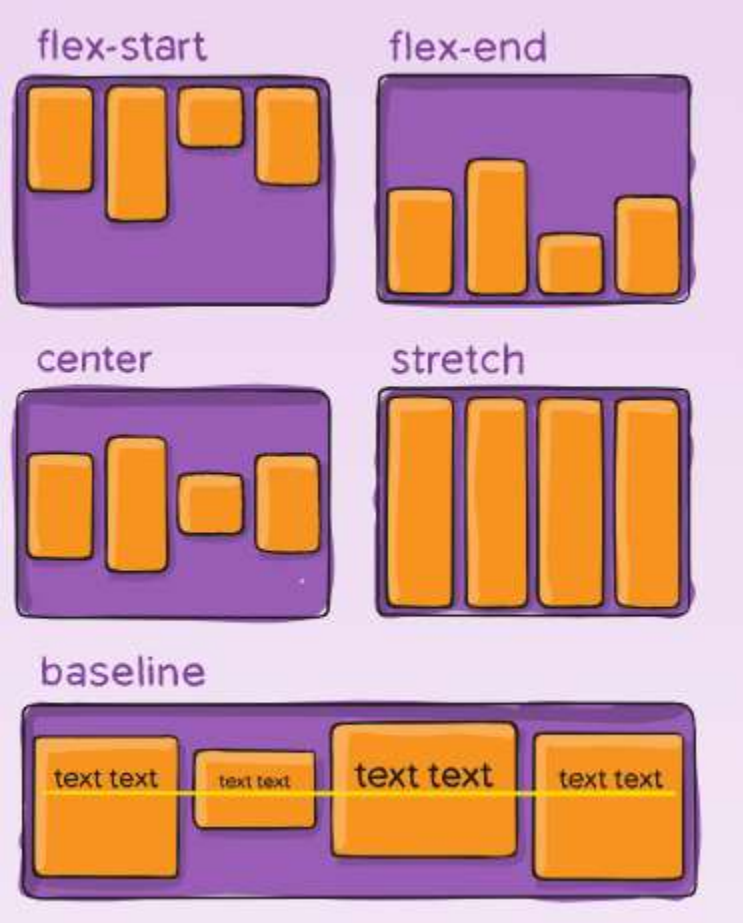
- flex-start (default): items are packed toward the start of the flex-direction.
- flex-end: items are packed toward the end of the flex-direction.
- start: items are packed toward the start of the writing-mode direction.
- end: items are packed toward the end of the writing-mode direction.
- left: items are packed toward left edge of the container, unless that doesn't make sense with the flex-direction, then it behaves like start.
- right: items are packed toward right edge of the container, unless that doesn't make sense with the flex-direction, then it behaves like end.
- center: items are centered along the line
- space-between: items are evenly distributed in the line; first item is on the start line, last item on the end line
- space-around: items are evenly distributed in the line with equal space around them. Note that visually the spaces aren't equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies.
- space-evenly: items are distributed so that the spacing between any two items (and the space to the edges) is equal.

align-items

This defines the default behavior for how flex items are laid out along the **cross axis** on the current line. Think of it as the justify-content version for the cross-axis (perpendicular to the main-axis).

For M

AL-JUNAID INSTITUTE OF GROUP




```
.container {
  align-items: stretch | flex-start | flex-end | center | baseline | first baseline | last baseline | start | end | self-start | self-end + ... safe | unsafe;
}
```

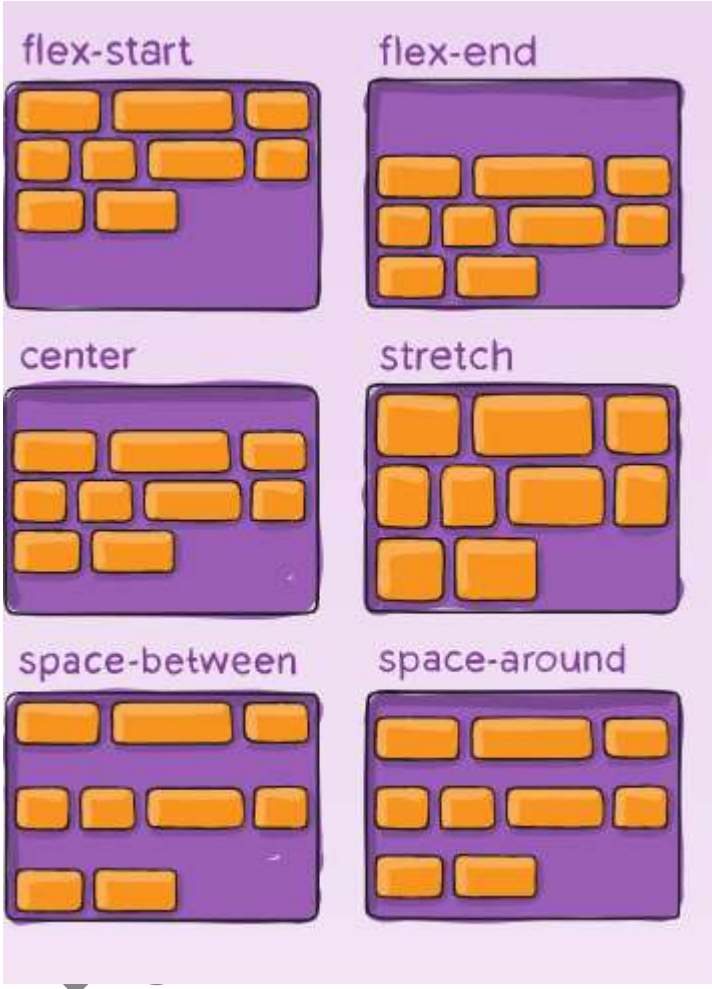
- stretch (default): stretch to fill the container (still respect min-width/max-width)
- flex-start / start / self-start: items are placed at the start of the cross axis. The difference between these is subtle, and is about respecting the flex-direction rules or the writing-mode rules.
- flex-end / end / self-end: items are placed at the end of the cross axis. The difference again is subtle and is about respecting flex-direction rules vs. writing-mode rules.

For More Visit: ...nshelp.com

- center: items are centered in the cross-axis
- baseline: items are aligned such as their baselines align

align-content

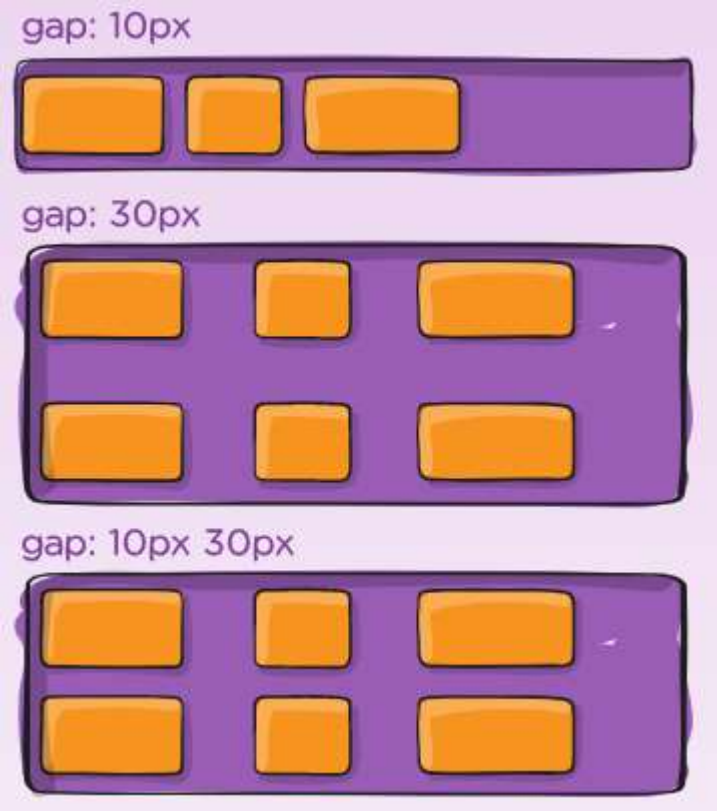
This aligns a flex container's lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis.




```
.container {
  align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline | last baseline + ... safe | unsafe;
}
```

- normal (default): items are packed in their default position as if no value was set.
- flex-start / start: items packed to the start of the container. The (more supported) flex-start honors the flex-direction while start honors the writing-mode direction.
- flex-end / end: items packed to the end of the container. The (more support) flex-end honors the flex-direction while end honors the writing-mode direction.
- center: items centered in the container
- space-between: items evenly distributed; the first line is at the start of the container while the last one is at the end
- space-around: items evenly distributed with equal space around each line
- space-evenly: items are evenly distributed with equal space around them
- stretch: lines stretch to take up the remaining space

gap, row-gap, column-gap



AL-JUNAID INSTITUTE OF GROUP

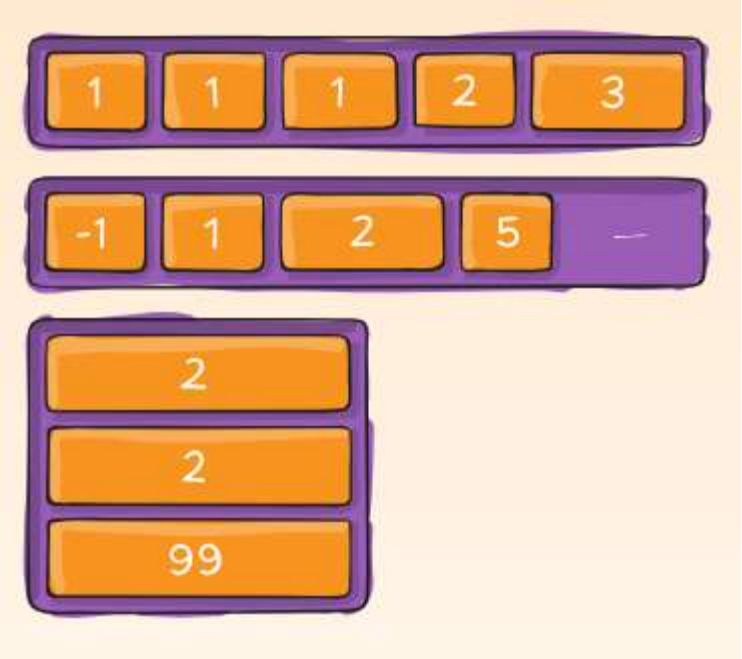
```
.container {
  display: flex;
   gap: 10px;
  gap: 10px 20px; /* row-gap column gap */
  row-gap: 10px;
  column-gap: 20px;
}
```

The behavior could be thought of as a minimum gutter, as if the gutter is bigger somehow (because of something like justify-content: space-between;) then the gap will only take effect if that space would end up smaller.

It is not exclusively for flexbox, gap works in grid and multi-column layout as well.

Properties for the Children
(flex items)

Order

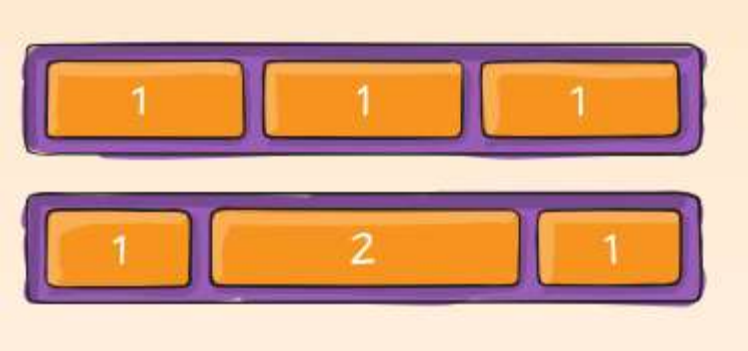


AL-JUNAID INSTITUTE OF GROUP

By default, flex items are laid out in the source order. However, the order property controls the order in which they appear in the flex container.

```
.item {
  order: 5; /* default is 0 */
}
```

flex-grow



This defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.

If all items have flex-grow set to 1, the remaining space in the container will be distributed equally to all children. If one of the children has a value of 2, that child would take up twice as much of the space either one of the others (or it will try, at least).

```
.item {
  flex-grow: 4; /* default 0 */
}
```

AL-JUNAID INSTITUTE OF GROUP

flex-shrink

This defines the ability for a flex item to shrink if necessary.

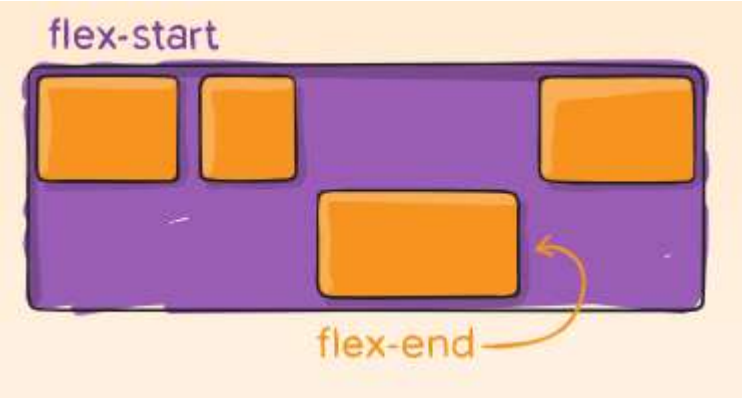
```
.item {
  flex-shrink: 3; /* default 1 */
}
```

flex-basis

This defines the default size of an element before the remaining space is distributed. It can be a length (e.g. 20%, 5rem, etc.) or a keyword.
The auto keyword means “look at my width or height property” (which was temporarily done by the main-size keyword until deprecated).
The content keyword means “size it based on the item’s content” – this keyword isn’t well supported yet, so it’s hard to test and harder to know what its brethren max-content, min-content, and fit-content do.

```
.item {
  flex-basis:  | auto; /* default auto */
}
```

align-self



his allows the default alignment (or the one specified by align-items) to be overridden for individual flex items.

```
.item {
  align-self: auto | flex-start | flex-end | center | baseline
| stretch;
}
```

For Moe Visit: vulmshelp.com