Float

The CSS `float` property specifies how an element should float.

The `float` property can have one of the following values:

- `left` - The element floats to the left of its container
- `right` - The element floats to the right of its container
- `none` - The element does not float (will be displayed just where it occurs in the text). This is default
- `inherit` - The element inherits the float value of its parent

Clearing floats

The CSS `clear` property specifies what elements can float beside the cleared element and on which side.

The `clear` property accepts the following values:

- `left`: Clear items floated to the left.
- `right`: Clear items floated to the right.
- `both`: Clear any floated items, left or right.

Using CSS custom properties (variables)

The `var()` function is used to insert the value of a CSS variable.

CSS variables can have a global or local scope.

To create a variable with global scope, declare it inside the `:root` selector. The `:root` selector matches the document's root element.

```
:root {
  --black: #000000;
  --white: #ffffff;
}
```

var(--*name*, *value*)

CSS Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

- `translate() -` moves an element from its current position (according to the parameters given for the X-axis and the Y-axis)
- `rotate() -` rotates an element clockwise or counter-clockwise according to a given degree
- `rotateX()`
- `rotateY()`
- `rotateZ()`
- `scaleX()`
- `scaleY()`
- `scale()`
- `skewX()`
- `skewY()`
- `skew()`
- `matrix()`

CSS Flexbox

- Flexible box, commonly referred to as flexbox, is a new layout model introduced in CSS3 for creating the flexible user interface design with multiple rows and columns without using the percentage or fixed length values.

- The CSS3 flex layout model provides a simple and powerful mechanism for handling the distributing of space and alignment of content automatically through stylesheet without interfering the actual markup.

Properties for theParent

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

```
.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.

- 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 top to bottom
- column-reverse: same as row-reverse but bottom to top

flex-wrap

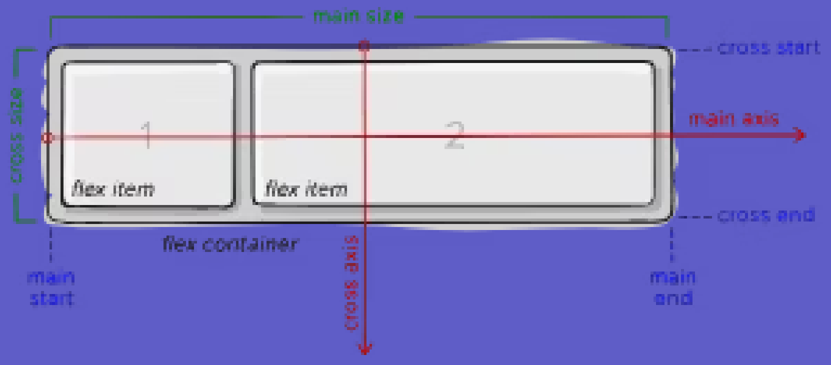
By default, flex items will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property.

- 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.

flex-flow (Applies to: parent flex container element)



- 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 nowrap.

order

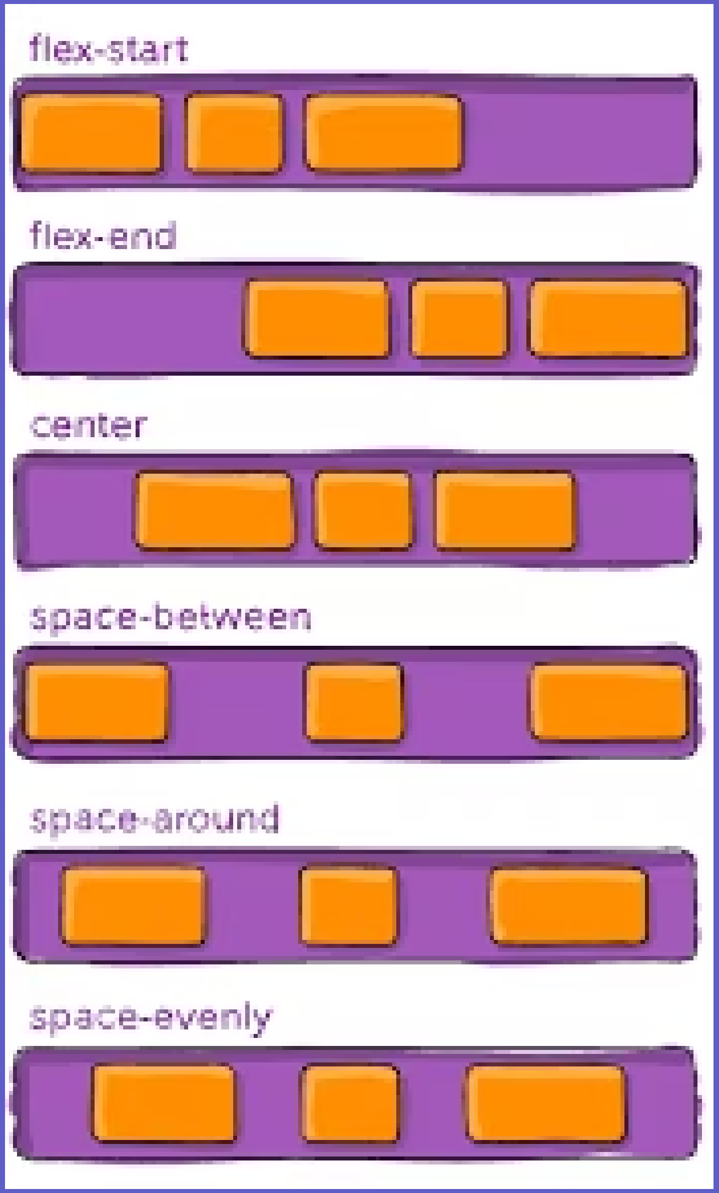
- By default, flex items are laid out in the source order.
- However, the order property controls the order in which they appear in the flex container.

Alignment

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 areflexible but have reached their maximum size.
- It also exerts some control over the alignment of items when they overflow the line.

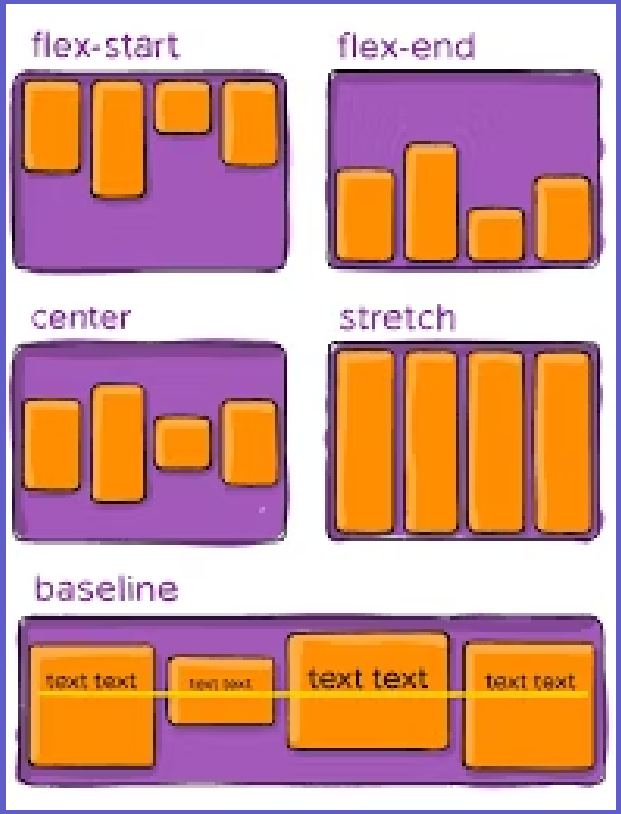
- **flex-start (default)**: items are packed toward the start of the flex-direction.
- **flex-end**: items are packed toward the end of the flex-direction.
- **center**: items are centered along the line
- **space-between**: items are evenly distributed in the line; first item is on the start line, last 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).

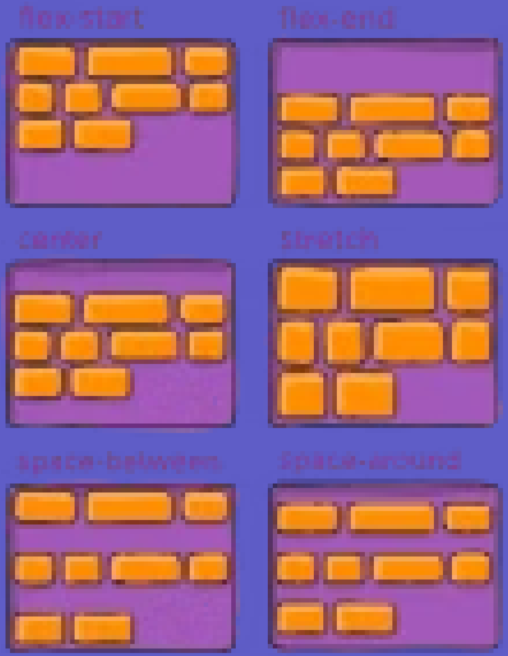
- stretch (default): stretch to fill the container (still respect min-width/max-width)
- flex-start / start / self-start: items are placed at the start of the cross axis.The difference between these is subtle, and is about respecting the flex-direction rules or the writing-mode rules.
- flex-end / end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

- 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 (default): lines stretch to take up the remaining space



Properties for the Children (flex items)

align-self

- This allows the default alignment (or the one specified by align-items) to be overridden for individual flex items.
- Please see the align-items explanation to understand the available values.
- Note that float, clear and vertical-align have no effect on a flex item.

flex-basis

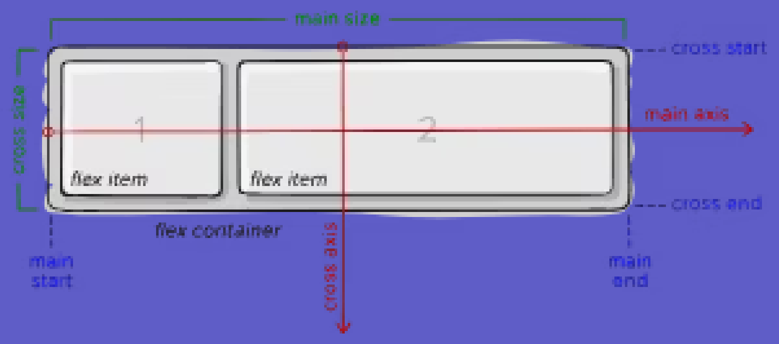
A flex-basis value set to `auto` sizes the element according to its size property (which can itself be the keyword `auto`, which sizes the element based on its contents).

flex-grow

- This defines the ability for a flex item to grow if necessary. It accepts a unit less 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, the remaining space would take up twice as much space as the others (or it will try to, at least).

flex-shrink

- This defines the ability for a flex item to shrink if necessary. (Negative numbers are invalid.)


main size
cross start
cross size
1
2
main axis
flex item
flex item
cross end
flex container
cross axis
main start
main end