

FLEXBOX OVERVIEW

Main & Cross Axis

The main axis is defined by the flex-direction property, and the cross axis runs perpendicular to it. These axes run vertical and horizontal and have 4 possible values.

row | row-reverse | column | column-reverse

Row

Choose row or row-reverse and your main axis will run along the row in the inline direction. The cross axis will run vertically or block direction.

Column

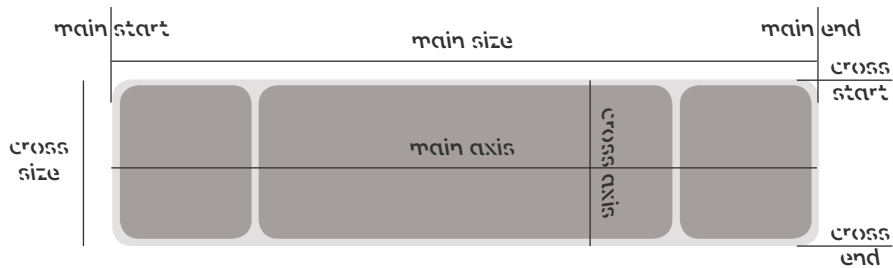
Choose column or column-reverse and your main axis will run from the top of the page to the bottom — in the block direction.

Cross Axis

The cross axis runs perpendicular to the main axis, therefore if your flex-direction (main axis) is set to row or row-reverse the cross axis runs down the columns.

row | row-reverse | column | column-reverse

Start & End Lines



Flex Container

The parent element known as the “flex container”. The direct children of the flex container become flex items.

Flex Items

Once a flex container is set, flex items will all line up in a row, using the size of the content as their size in the main axis.

Flexbox vs. Grid

Flexbox is a one-dimensional layout method whereas Grid Layout is a two-dimensional layout method.

Grid or Flexbox?

As a rule of thumb, if you are adding widths to flex items in order to make items in one row of a wrapped flex container line up with the items above them you really want two-dimensional layout.

Grid Layout

In Grid Layout you do the majority of sizing specification on the container, setting up tracks and then placing items into them.

Flex Layout

Flex items can be allowed to wrap but, once they do so, each line becomes a flex container of its own.

In flexbox, while you create a flex container and set the direction at that level, any control over item sizing needs to happen on the items themselves.

Writing Modes

Modern layout methods encompass a range of writing modes and we cannot assume that a line of text starts at the top left of a document and run towards the right hand side, with new lines appearing one under the other.

Flex layout is based on “flex-flow directions”.

FLEXBOX PROPERTIES & VALUES

flex

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

`display: flex`

flex-direction

Specifies how flex items are placed in the flex container, by setting the direction of the flex container's main axis. This determines the direction in which flex items are laid out.

row | row-reverse | column | column-reverse

flex-wrap

Controls whether the flex container is single-line or multi-line, and the direction of the cross-axis, which determines the direction new lines are stacked in.

`nowrap | wrap | wrap-reverse;`

flex-flow

Shorthand for flex-direction and flex-wrap properties, which together define the flex container's main and cross axes. Default is row nowrap.

`flex-flow: <'flex-direction'> || <'flex-wrap'>`

justify-content

Aligns flex items along the main axis of the current line of the flex container. Typically 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.

`flex-start` | `flex-end` | `center` | `space-between` | `space-around` | `space-evenly`;

align-items

Defines the default behavior for how flex items are laid out along the cross axis on the current line.

`stretch | flex-start | flex-end | center | baseline;`

align-content

aligns a flex container's lines within the flex container when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis. Note, this property has no effect on a single-line flex container.

flex-start | flex-end | center | space-between | space-around | stretch;

order

Flex items are, by default, displayed and laid out in the same order as they appear in the source document. The `order` property can be used to change this ordering.

```
order: <integer>; /* default is 0 */
```

flex-grow

Determines how much the flex item will grow relative to the rest of the flex items in the flex container when positive free space is distributed.

When omitted, it is set to 1.

flex-shrink

Determines how much the flex item will shrink relative to the rest of the flex items in the flex container when negative free space is distributed. When omitted, it is set to 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.

```
flex-basis: <length> | auto; /* default auto */
```

flex

This is the shorthand for flex-grow, flex-shrink and flex-basis combined. The second and third parameters (flex-shrink and flex-basis) are optional. Default is 0 1 auto. The short hand sets the other values intelligently.

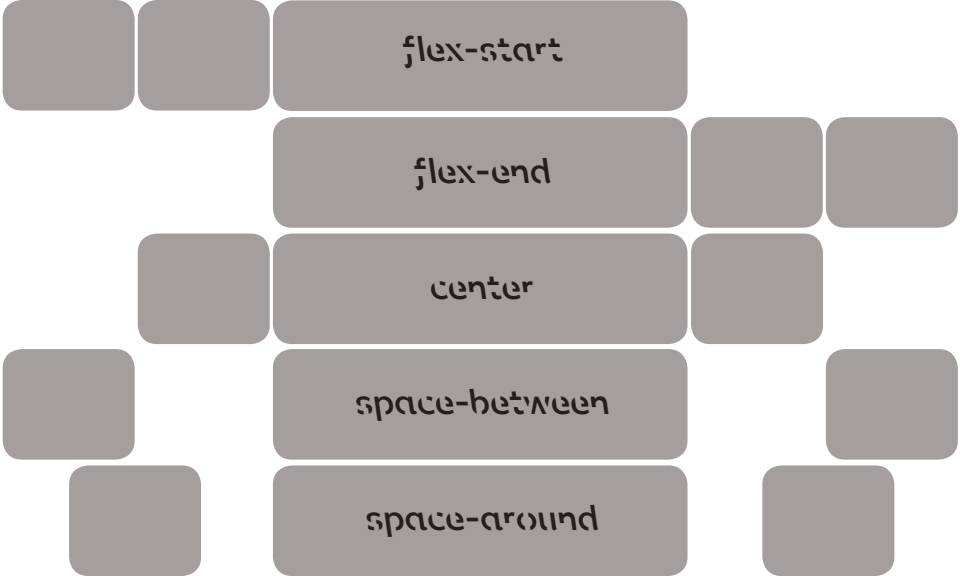
`flex: none | [<'flex-grow'> <'flex-shrink'>? || <'flex-basis'>]`

align-self

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

`align-self: auto | flex-start | flex-end | center | baseline | stretch;`

**justify-content
values**



A diagram consisting of a 5x5 grid of squares. A central path of five squares is highlighted, each containing text. The path starts at the top center, moves down to the middle center, then left to the middle-left, then down to the bottom-left, and finally right to the bottom center. The text in the squares, from top to bottom, is: flex-start, flex-end, center, space-between, and space-around. The squares not on this path are empty.

flex-start


flex-end

center

space-between


space-around

**align-items:
flex-start**



flex-start

**align-items:
flex-end**

The image features four gray rounded rectangles on a white background. From left to right: a tall rectangle, a short rectangle, a wide rectangle, and another tall rectangle. The text 'flex-end' is centered within the wide rectangle.

flex-end

**align-items:
center**



center

**align-items:
stretch**

stretch

**align-items:
stretch**

stretch

**align-items:
baseline**



all the

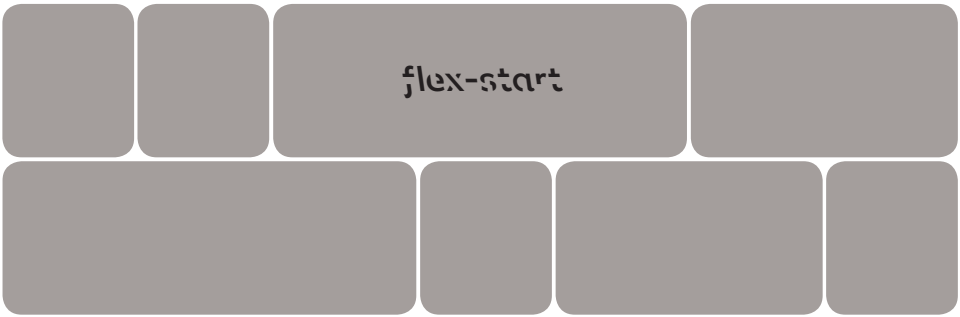
words

are on the same

line

baseline

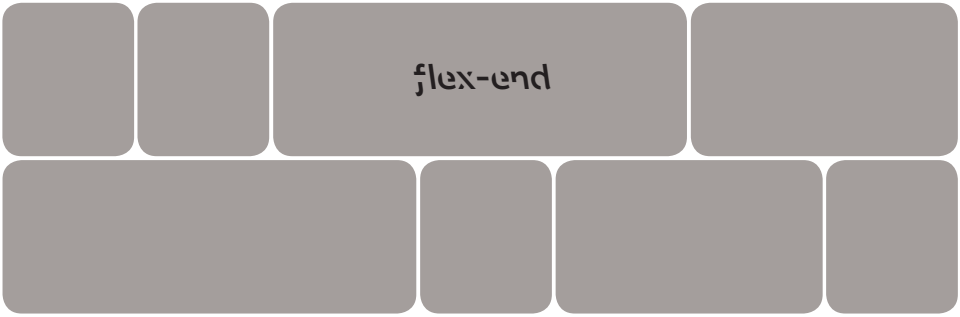
**align-content:
flex-start**



A diagram illustrating the 'flex-start' alignment in a flex container. The container is represented by a horizontal row of four gray rounded rectangles. The first three rectangles are of equal width and height, while the fourth is wider. The text 'flex-start' is centered above the first three rectangles. Below the row, there is a large white space.

flex-start

**align-content:
flex-end**

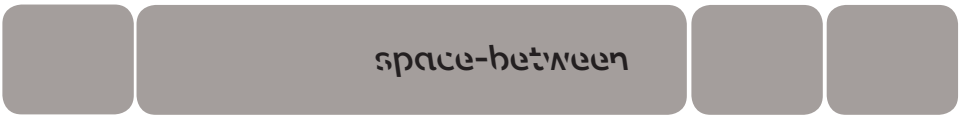


flex-end

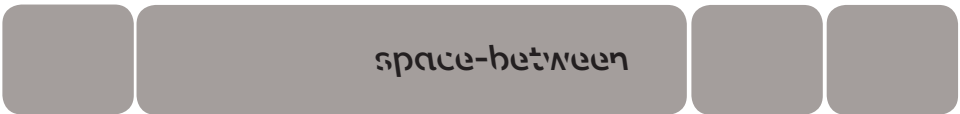
**align-content:
center**

center

**align-content:
space-between**




**align-content:
space-around**



**align-content:
stretch**

stretch

**justify-content:
flex-start**



`flex-start`

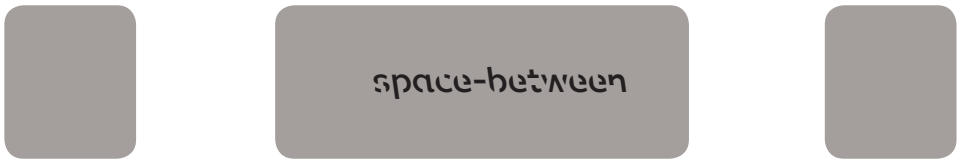
**justify-content:
flex-end**



justify-content:
center

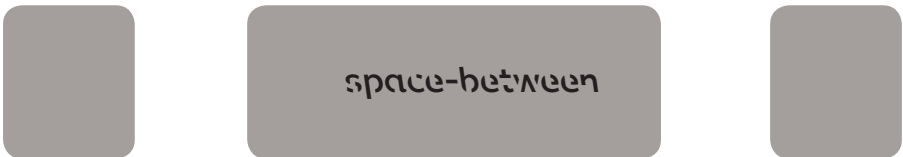


**justify-content:
space-between**



space-between

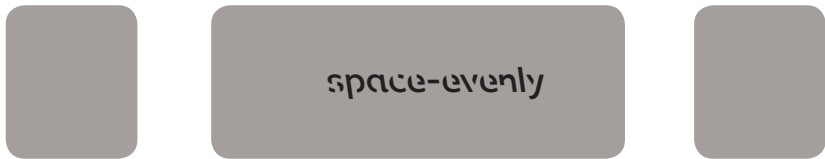
**justify-content:
space-around**



space-between

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.

**justify-content:
space-evenly**



items are distributed so that the spacing between any two items (and the space to the edges) is equal.