Change theme

display

w3.org/TR/css-flexbox-1/#flex-containers

flex | inline-flex

Some text

1 2 3 4

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

A flex container establishes a new ***flex formatting context*** for its contents. This is the same as establishing a block formatting context, except that flex layout is used instead of block layout. For example, floats do not intrude into the flex container, and the flex container's margins do not collapse with the margins of its contents. Flex containers form a containing block for their contents exactly like block containers do. [CSS21] The overflow property applies to flex containers.

Flex containers are not block containers, and so some properties that were designed with the assumption of block layout don't apply in the context of flex layout. In particular:

- float and clear do not create floating or clearance of flex item, and do not take it out-of-flow.
- vertical-align has no effect on a flex item.
- the ::first-line and ::first-letter pseudo-elements do not apply to flex containers, and flex containers do not contribute a first formatted line or first letter to their ancestors.

If an element's specified display is inline-flex, then its display property computes to flex in certain circumstances: the table in CSS 2.1 Section 9.7 is amended to contain an additional row, with inline-flex in the "Specified Value" column and flex in the "Computed Value" column.

Applies to: all elements.

Values

flex

This value causes an element to generate a flex container box that is block-level when placed in flow layout.

inline-flex

This value causes an element to generate a flex container box that is inline-level when placed in flow layout.

Ordering & Orientation

https://www.w3.org/TR/css-flexbox-1/#flow-order

The contents of a flex container can be laid out in any direction and in any order. This allows an author to trivially achieve effects that would previously have required complex or fragile methods, such as hacks using the float and clear properties. This functionality is exposed through the flex-direction, flex-wrap, and order properties.

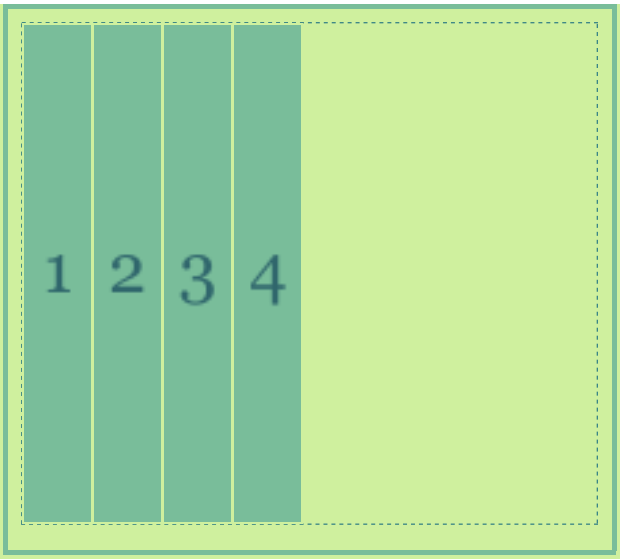
> Note: The reordering capabilities of flex layout intentionally affect *only the visual rendering*, leaving speech order and navigation based on the source order. This allows authors to manipulate the visual presentation while leaving the source order intact for non-CSS UAs and for linear models such as speech and sequential navigation. See Reordering and Accessibility and the Flex Layout Overview for examples that use this dichotomy to improve accessibility.

Authors *must not* use order or the `*-reverse` values of flex-flow/flex-direction as a substitute for correct source ordering, as that can ruin the accessibility of the document.

flex-direction

w3.org/TR/css-flexbox-1/#flex-direction-property

row | row-reverse | column | column-reverse




```
.parent {
  display: flex;
  flex-direction: row;
  height: 250px;
}
```

The flex-direction property 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.

> Note: The reverse values do not reverse box ordering: like writing-mode and direction [CSS3-WRITING-MODES], they only change the direction of flow. Painting order, speech order, and sequential navigation orders are not affected.

Applies to: flex containers.

Initial: row.

Values

row

The flex container's main axis has the same orientation as the inline axis of the current writing mode. The main-start and main-end directions are equivalent to the inline-start and inline-end directions, respectively, of the current writing mode.

row-reverse

Same as row, except the main-start and main-end directions are swapped.

column

The flex container's main axis has the same orientation as the block axis of the current writing mode. The main-start and main-end directions are equivalent to the block-start and block-end directions, respectively, of the current writing mode.

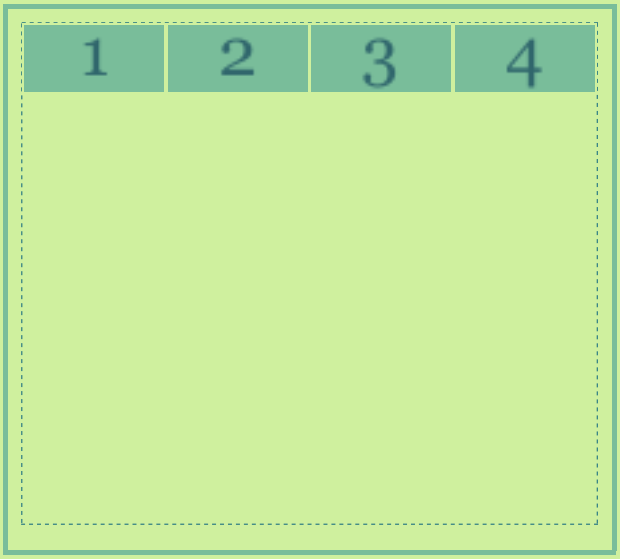
column-reverse

Same as column, except the main-start and main-end directions are swapped.

flex-wrap

w3.org/TR/css-flexbox-1/#flex-wrap-property

nowrap | wrap | wrap-reverse




```
.parent {
  display: flex;
  align-items: flex-start;
  flex-wrap: nowrap;
  height: 250px;
}

.child {
  width: 40%;
}
```

The flex-wrap property 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.

For the values that are not wrap-reverse, the cross-start direction is equivalent to either the inline-start or block-start direction of the current writing mode (whichever is in the cross axis) and the cross-end direction is the opposite direction of cross-start. When flex-wrap is wrap-reverse, the cross-start and cross-end directions are swapped.

Applies to: flex containers.

Initial: nowrap.

Values

nowrap

The flex container is single-line.

wrap

The flex container is multi-line.

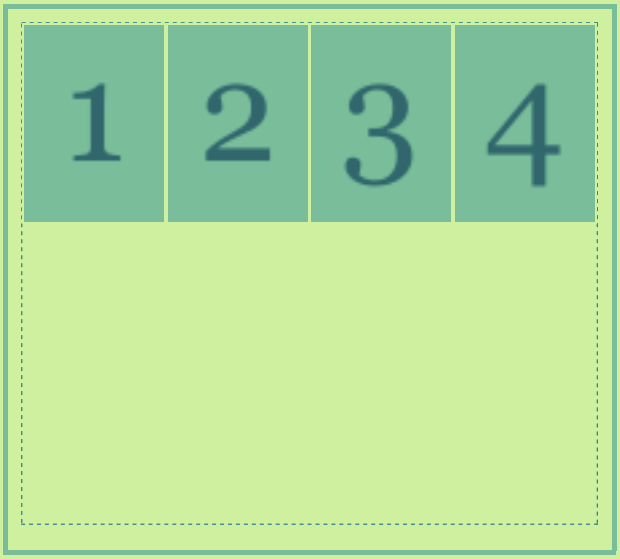
wrap-reverse

Same as wrap.

flex-flow

w3.org/TR/css-flexbox-1/#flex-flow-property

row nowrap | column-reverse | column wrap | row-reverse wrap-reverse




```
.parent {
  display: flex;
  flex-flow: row nowrap;
  height: 250px;
}

.child {
  width: 40%;
  height: 40%;
}
```

The flex-flow property is a shorthand for setting the flex-direction and flex-wrap properties, which together define the flex container's main and cross axes.

> Note that the flex-flow directions are writing mode sensitive. In vertical Japanese, for example, a row flex container lays out its contents from top to bottom.

Applies to: flex containers.

Initial: row nowrap.

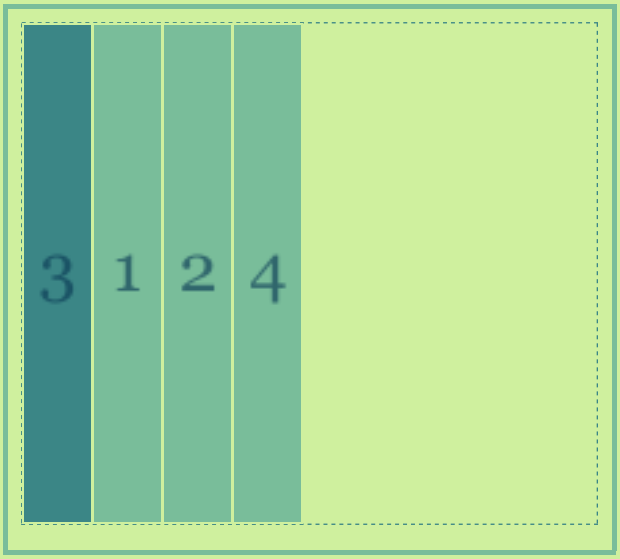
Value

<flex-direction> || **<flex-wrap>**

order

https://www.w3.org/TR/css-flexbox-1/#order-property

-1 0 1




```
.parent {
  display: flex;
  height: 250px;
}

.child--featured {
  order: -1;
}
```

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.

The order property controls the order in which flex items appear within the flex container, by assigning them to ordinal groups. It takes a single ***<integer>*** value, which specifies which ordinal group the flex item belongs to.

A flex container lays out its content in ***order-modified document order***, starting from the lowest numbered ordinal group and going up. Items with the same ordinal group are laid out in the order they appear in the source document. This also affects the painting order [CSS21], exactly as if the flex items were reordered in the source document. Absolutely-positioned children of a flex container are treated as having order: 0 for the purpose of determining their painting order relative to flex items.

Unless otherwise specified by a future specification, this property has no effect on boxes that are not flex items.

Applies to: flex items.

Initial: 0.

Value

<integer>

Alignment

https://www.w3.org/TR/css-flexbox-1/#alignment

After a flex container's contents have finished their flexing and the dimensions of all flex items are finalized, they can then be aligned within the flex container.

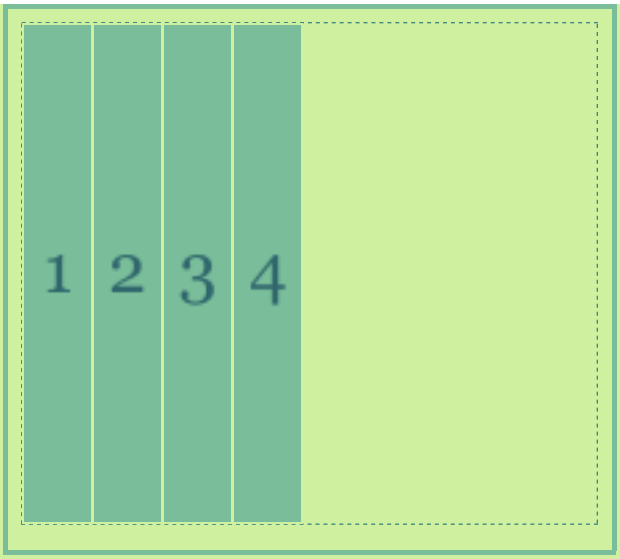
The margin properties can be used to align items in a manner similar to, but more powerful than, what margins can do in block layout. Flex items also respect the alignment properties from CSS Box Alignment, which allow easy keyword-based alignment of items in both the main axis and cross axis. These properties make many common types of alignment trivial, including some things that were very difficult in CSS 2.1, like horizontal and vertical centering.

> Note: While the alignment properties are defined in CSS Box Alignment [CSS-ALIGN-3], Flexible Box Layout reproduces the definitions of the relevant ones here so as to not create a normative dependency that may slow down advancement of the spec. These properties apply only to flex layout until CSS Box Alignment Level 3 is finished and defines their effect for other layout modes. Additionally, any new values defined in the Box Alignment module will apply to Flexible Box Layout; in otherwords, the Box Alignment module, once completed, will supercede the definitions here.

justify-content

w3.org/TR/css-flexbox-1/#justify-content-property

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




```
.parent {
  display: flex;
  justify-content: flex-start;
  height: 250px;
}
```

The justify-content property aligns flex items along the main axis of the current line of the flex container. This is done *after* any flexible lengths and any auto margins have been resolved. 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. It also exerts some control over the alignment of items when they overflow the line.

Applies to: flex containers.

Initial: flex-start.

Values

flex-start

Flex items are packed toward the start of the line. The main-start margin edge of the first flex item on the line is placed flush with the main-start edge of the line, and each subsequent flex item is placed flush with the preceding item.

flex-end

Flex items are packed toward the end of the line. The main-end margin edge of the last flex item is placed flush with the main-end edge of the line, and each preceding flex item is placed flush with the subsequent item.

center

Flex items are packed toward the center of the line. The flex items on the line are placed flush with each other and aligned in the center of the line, with equal amounts of space between the main-start edge of the line and the first item on the line and between the main-end edge of the line and the last item on the line. (If the leftover free-space is negative, the flex items will overflow equally in both directions.)

space-between

Flex items are evenly distributed in the line. If the leftover free-space is negative or there is only a single flex item on the line, this value is identical to flex-start.

Otherwise, the main-start margin edge of the first flex item on the line is placed flush with the main-start edge of the line, the main-end margin edge of the last flex item on the line is placed flush with the main-end edge of the line, and the remaining flex items on the line are distributed so that the spacing between any two adjacent items is the same.

space-around

Flex items are evenly distributed in the line, with half-size spaces on either end. If the leftover free-space is negative or there is only a single flex item on the line, this value is identical to center. Otherwise, the flex items on the line are distributed such that the spacing between any two adjacent flex items on the line is the same, and the spacing between the first/last flex items and the flex container edges is half the size of the spacing between flex items.

space-evenly

The alignment subjects are evenly distributed in the alignment container, with a full-size space on either end. The alignment subjects are distributed so that the spacing between any two adjacent alignment subjects, before the first alignment subject, and after the last alignment subject is the same.

The default fallback alignment for this value is center.

align-items, align-self

https://www.w3.org/TR/css-flexbox-1/#align-items-property

Flex items can be aligned in the cross axis of the current line of the flex container, similar to justify-content but in the perpendicular direction. align-items sets the default alignment for all of the flex container's items, including anonymous flex items. align-self allows this default alignment to be overridden for individual flex items. (For anonymous flex items, align-self always matches the value of align-items on their associated flex container.)

If either of the flex item's cross-axis margins are auto, align-self has no effect.

Values

auto

Defers cross-axis alignment control to the value of align-items on the parent box. (This is the initial value of align-self.)

flex-start

The cross-start margin edge of the flex item is placed flush with the cross-start edge of

the line.

flex-end

The cross-end margin edge of the flex item is placed flush with the cross-end edge of the line.

center

The flex item's margin box is centered in the cross axis within the line. (If the cross size of the flex line is less than that of the flex item, it will overflow equally in both directions.)

baseline

The flex item ***participates in baseline alignment***: all participating flex items on the line are aligned such that their baselines align, and the item with the largest distance between its baseline and its cross-start margin edge is placed flush against the cross-start edge of the line. If the item does not have a baseline in the necessary axis, then one is synthesized from the flex item's border box.

stretch

If the cross size property of the flex item computes to auto, and neither of the cross-axis margins are `auto`, the flex item is ***stretched***. Its used value is the length necessary to make the cross size of the item's margin box as close to the same size as the line as possible, while still respecting the constraints imposed by min-height/min-width/max-height/max-width.

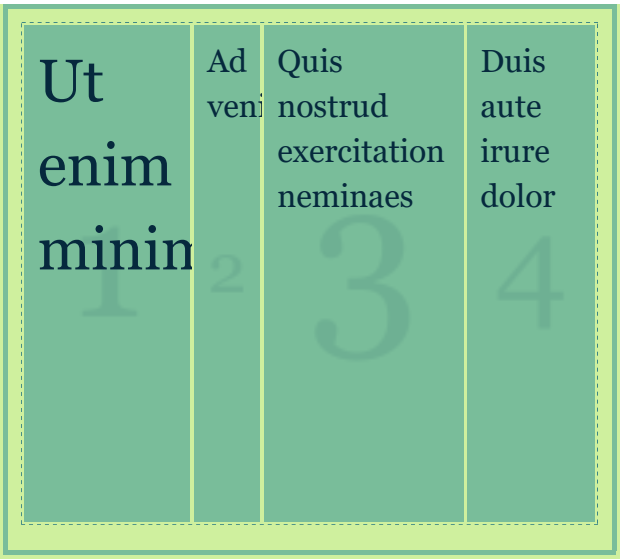
> Note: If the flex container's height is constrained this value may cause the contents of the flex item to overflow the item.

The cross-start margin edge of the flex item is placed flush with the cross-start edge of the line.

align-items

w3.org/TR/css-flexbox-1/#propdef-align-items

stretch | flex-start | flex-end | center | baseline | auto




```
.parent {
  display: flex;
  align-items: stretch;
  height: 250px;
}
```

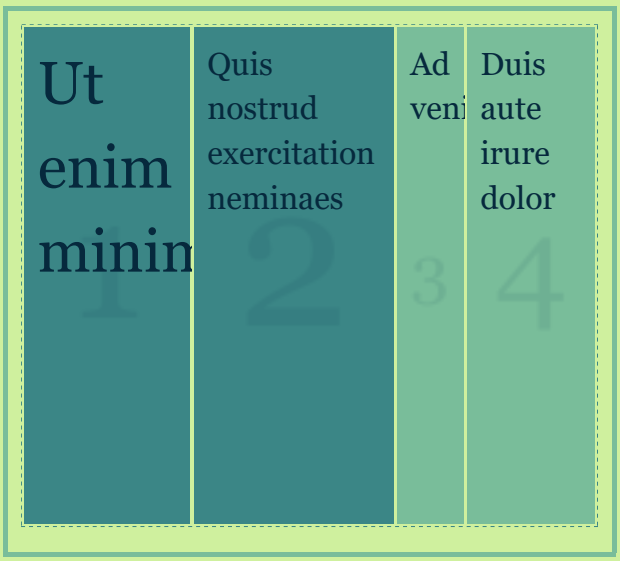
Applies to: flex containers.

Initial: stretch.

align-self

w3.org/TR/css-flexbox-1/#propdef-align-self

auto | stretch | flex-start | flex-end | center | baseline




```
.parent {
  display: flex;
  height: 250px;
}

.child--featured {
  align-self: auto;
}
```

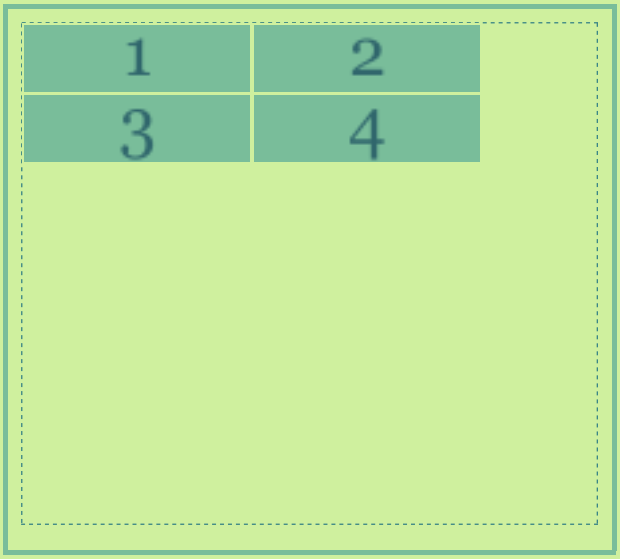
Applies to: flex items.

Initial: auto.

align-content

w3.org/TR/css-flexbox-1/#align-content-property

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




```
.parent {
  display: flex;
  flex-wrap: wrap;
  align-content: flex-start;
  height: 250px;
}

.child {
  width: 40%;
}
```

The align-content property 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. Values have the following meanings:

> Note: Only multi-line flex containers ever have free space in the cross-axis for lines to be aligned in, because in a single-line flex container the sole line automatically stretches to fill the space.

Applies to: multi-line flex containers.

Initial: stretch.

Values

flex-start

Lines are packed toward the start of the flex container. The cross-start edge of the first line in the flex container is placed flush with the cross-start edge of the flex container, and each subsequent line is placed flush with the preceding line.

flex-end

Lines are packed toward the end of the flex container. The cross-end edge of the last line is placed flush with the cross-end edge of the flex container, and each preceding

line is placed flush with the subsequent line.

center

Lines are packed toward the center of the flex container. The lines in the flex container are placed flush with each other and aligned in the center of the flex container, with equal amounts of space between the cross-start content edge of the flex container and the first line in the flex container, and between the cross-end content edge of the flex container and the last line in the flex container. (If the leftover free-space is negative, the lines will overflow equally in both directions.)

space-between

Lines are evenly distributed in the flex container. If the leftover free-space is negative or there is only a single flex line in the flex container, this value is identical to flex-start. Otherwise, the cross-start edge of the first line in the flex container is placed flush with the cross-start content edge of the flex container, the cross-end edge of the last line in the flex container is placed flush with the cross-end content edge of the flex container, and the remaining lines in the flex container are distributed so that the spacing between any two adjacent lines is the same.

space-around

Lines are evenly distributed in the flex container, with half-size spaces on either end. If the leftover free-space is negative this value is identical to center. Otherwise, the lines in the flex container are distributed such that the spacing between any two adjacent lines is the same, and the spacing between the first/last lines and the flex container edges is half the size of the spacing between flex lines.

stretch

Lines stretch to take up the remaining space. If the leftover free-space is negative, this value is identical to flex-start. Otherwise, the free-space is split equally between all of the lines, increasing their cross size.

Flexibility

https://www.w3.org/TR/css-flexbox-1/#flexibility

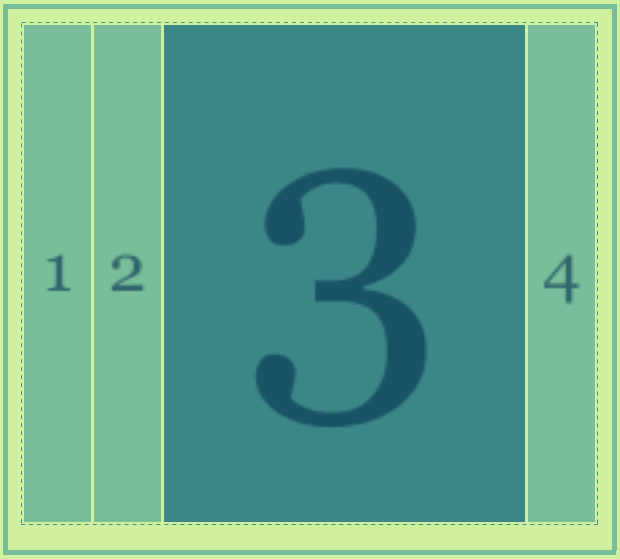
The defining aspect of flex layout is the ability to make the flex items “flex”, altering their width/height to fill the available space in the main dimension. This is done with the flex property. A flex container distributes free space to its items (proportional to their flex grow factor) to fill the container, or shrinks them (proportional to their flex shrink factor) to prevent overflow.

A flex item is ***fully inflexible*** if both its flex-grow and flex-shrink values are zero, and ***flexible*** otherwise.

flex-grow

w3.org/TR/css-flexbox-1/#flex-grow-property

0 1




```
.parent {
  display: flex;
  height: 250px;
}

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

The flex-grow property sets the flex grow factor to the provided ***<number>***. Negative numbers are invalid.

Applies to: flex items.

Initial: 0.

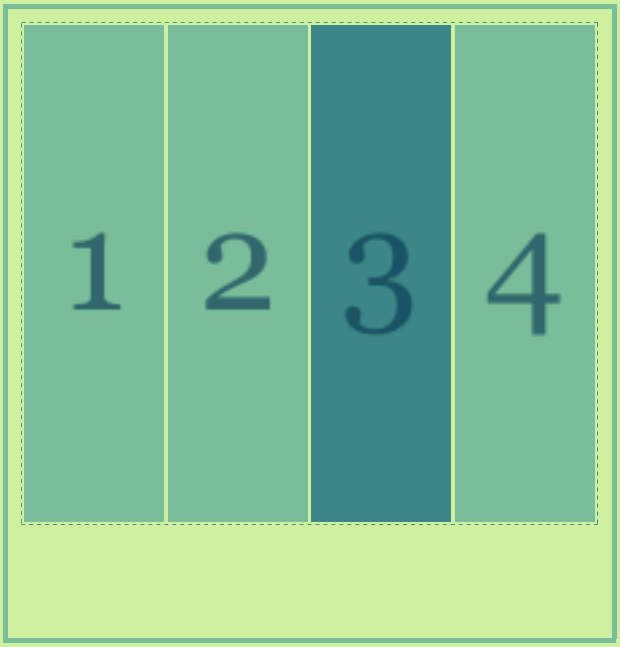
Value

<number>

flex-shrink

w3.org/TR/css-flexbox-1/#flex-shrink-property

1 0




```
.parent {
  display: flex;
  height: 250px;
}

.child {
  width: 45%;
}

.child--featured {
  flex-shrink: 1;
}
```

The flex-shrink property sets the flex shrink factor to the provided ***<number>***. Negative numbers are invalid.

Applies to: flex items.

Initial: 1.

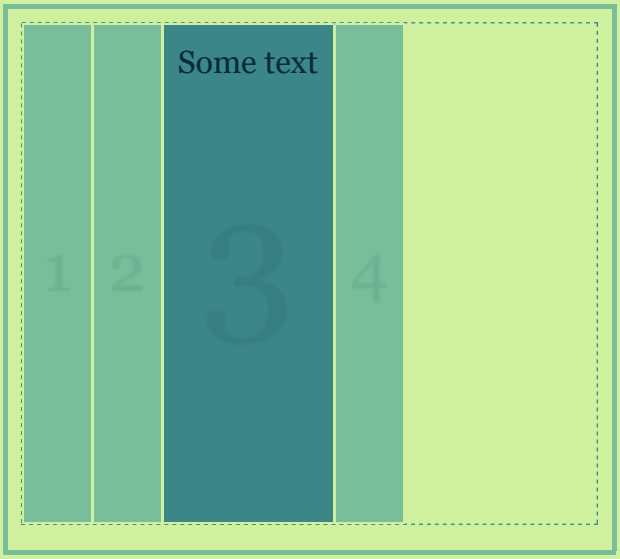
Value

<number>

flex-basis

w3.org/TR/css-flexbox-1/#flex-basis-property

30% 50% content auto




```
.parent {
  display: flex;
  flex-wrap: wrap;
  height: 250px;
}

.child--featured {
  flex-basis: 30%;
}
```

The flex-basis property sets the flex basis. It accepts the same values as the width and height property, plus content.

For all values other than auto and content (defined above), flex-basis is resolved the same way as width in horizontal writing modes [CSS21], except that if a value would resolve to auto for width, it instead resolves to content for flex-basis. For example, percentage values of flex-basis are resolved against the flex item's containing block (i.e. its flex container); and if that containing block's size is indefinite, the used value for flex-basis is content. As another corollary, flex-basis determines the size of the content box, unless otherwise specified such as by box-sizing [CSS3UI].

Applies to: flex items.

Initial: auto.

Value

content | **<width>**

flex

https://www.w3.org/TR/css-flexbox-1/#flex-property

The flex property specifies the components of a ***flexible length***: the ***flex factors*** (grow and shrink) and the flex basis. When a box is a flex item, flex is consulted *instead of* the main size property to determine the main size of the box. If a box is not a flex item, flex has no effect.

The initial values of the flex components are equivalent to flex: 0 1 auto.

> Note: The initial values of flex-grow and flex-basis are different from their defaults when omitted in the flex shorthand. This is so that the flex shorthand can better accommodate the most common cases.

A unitless zero that is not already preceded by two flex factors must be interpreted as a flex factor. To avoid misinterpretation or invalid declarations, authors must specify a zero <flex-basis> component with a unit or precede it by two flex factors.

Applies to: flex items.

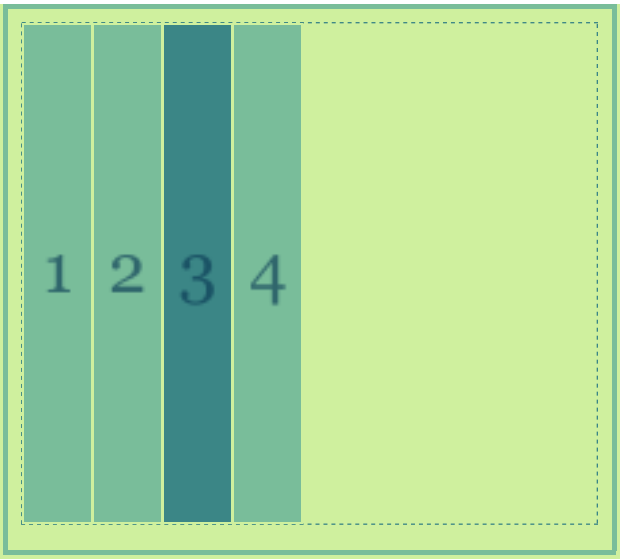
Initial: 0 1 auto.

Value

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

Flex values

0 1 auto | 0 1 30% | 1 0 30% | none




```
.parent {
  display: flex;
  height: 250px;
}

.child--featured {
  flex: 0 1 auto;
}
```

<flex-grow>

This <number> component sets flex-grow longhand and specifies the ***flex grow factor***, which 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 values between 0 and 1 have a somewhat special behavior: when the sum of the flex values on the line is less than 1, they will take up less than 100% of the free space.

<flex-shrink>

This <number> component sets flex-shrink longhand and specifies the ***flex shrink factor***, which 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`.

> Note: The flex shrink factor is multiplied by the flex base size when distributing negative space. This distributes negative space in proportion to how much the item is able to shrink, so that e.g. a small item won't shrink to zero before a larger item has been noticeably reduced.

<flex-basis>

This component sets the flex-basis longhand, which specifies the ***flex basis***: the initial main size of the flex item, before free space is distributed according to the flex factors.

<flex-basis> accepts the same values as the width and height properties (except that auto is treated differently) plus the content keyword:

auto

When specified on a flex item, the auto keyword retrieves the value of the main size property as the used flex-basis. If that value is itself auto, then the used value is content.

content

Indicates an automatic size based on the flex item's content. (It is typically equivalent to the max-content size, but with adjustments to handle aspect ratios, intrinsic sizing constraints, and orthogonal flows; see details in §9 Flex Layout Algorithm.)

> Note: This value was not present in the initial release of Flexible Box Layout, and thus some older implementations will not support it. The equivalent effect can be achieved by using `auto` together with a main size (width or height) of auto.

<width>

For all other values, flex-basis is resolved the same way as for width and height.

When omitted from the flex shorthand, its specified value is `0`.

none

The keyword none expands to `0 0 auto`.