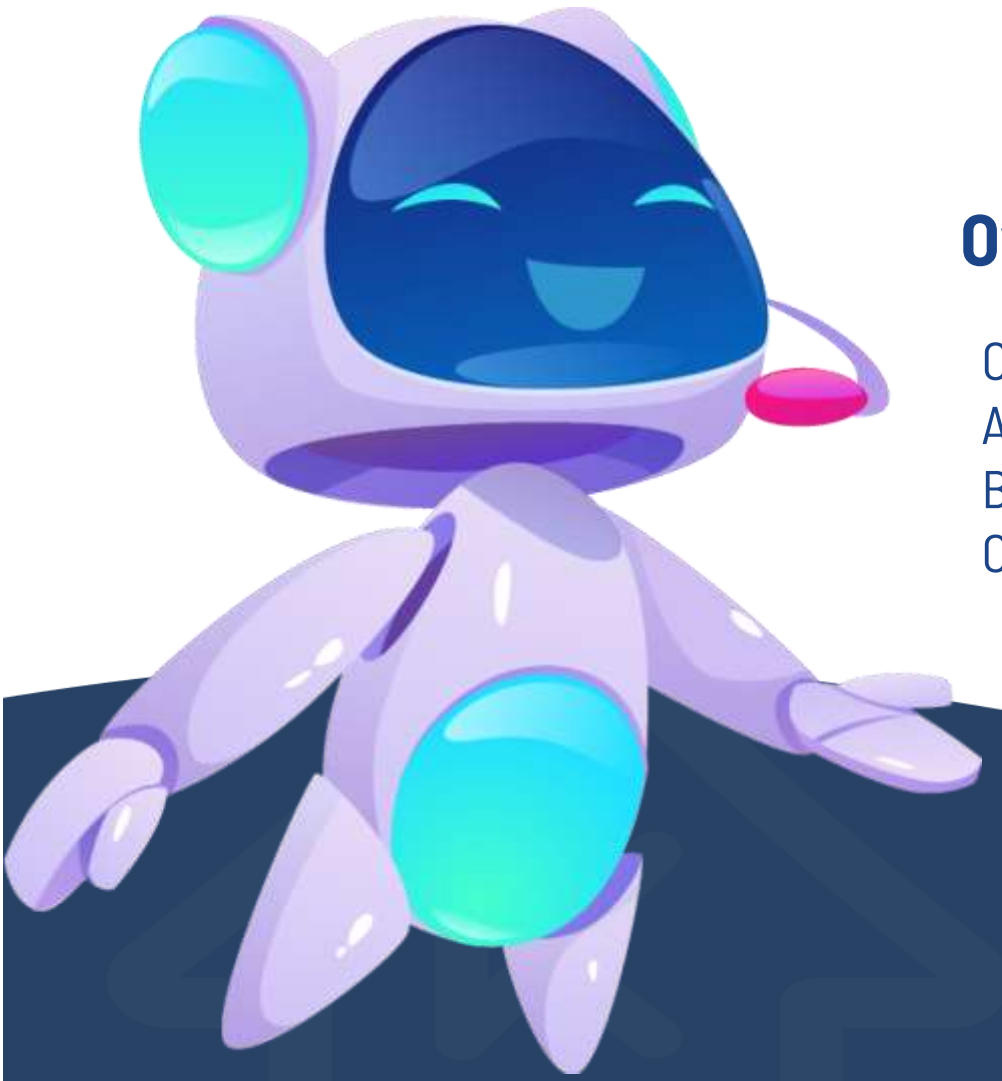

KodeGo



KodeGo

Overview

CSS FLEXBOX

A. Flexbox
B. Flexbox Properties
C. Browser Support

Flexbox

- sometimes called a flexible box layout module or flexbox layout module.
- It makes browsers display selected HTML elements as flexible box models.
- It allows easy resizing and repositioning of a flexible container and its items one-dimensionally.
- "One-dimensionally" means Flexbox allows laying out box models in a row or column at a time. In other words, it cannot lay out box models in a row and column at the same time.

CSS Flexbox Layout Module

KodeGo

Before the Flexbox Layout module, there were 4 layout modes:

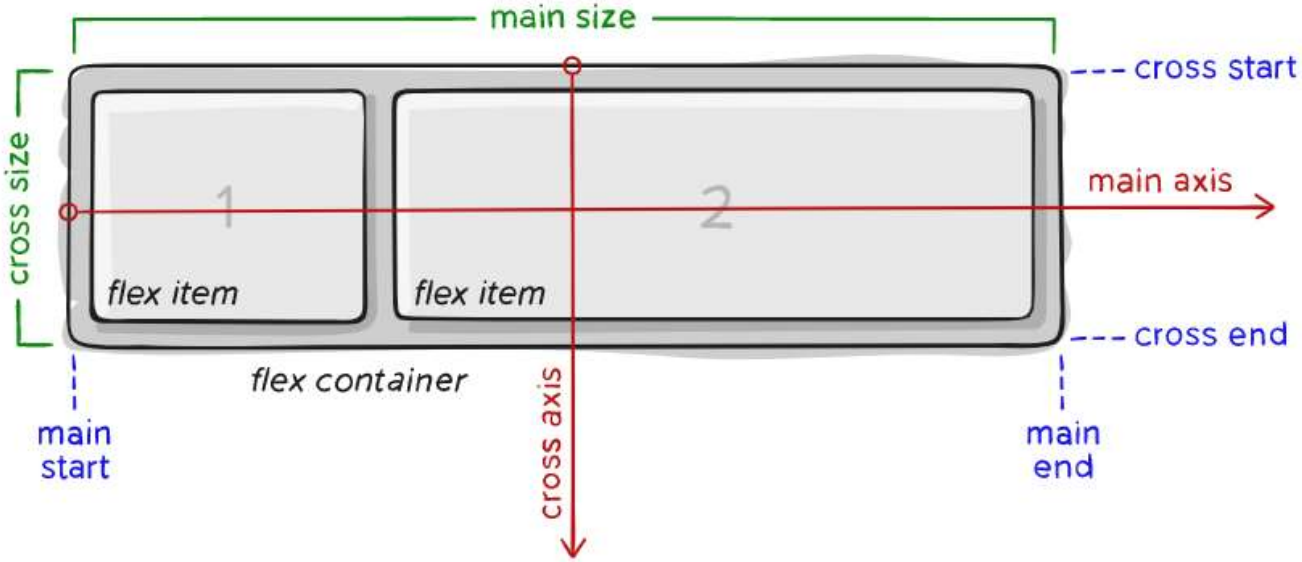
1. Block, for sections in a webpage
2. Inline, for text
3. Table, for two-dimensional table data
4. Positioned, for explicit position of an element

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

Flex Layout

KodeGo

- the flex layout is based on “flex-flow directions”

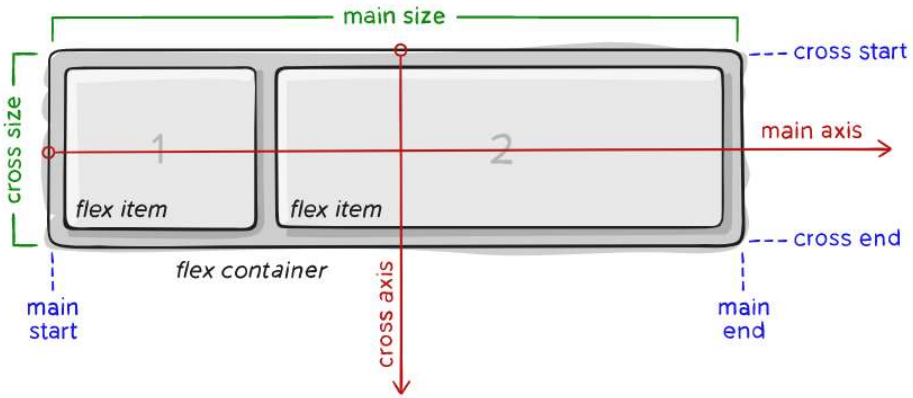


Flex Layout

main axis

– is the primary axis along which flex items are laid out.

- it is not necessarily horizontal; it depends on the flex-direction property.



main-start | main-end

– The flex items are placed within the container starting from main-start and going to main-end.

main size

– A flex item's width or height, whichever is in the main dimension, is the item's main size.

- The flex item's main size property is either the 'width' or 'height' property, whichever is in the main dimension.

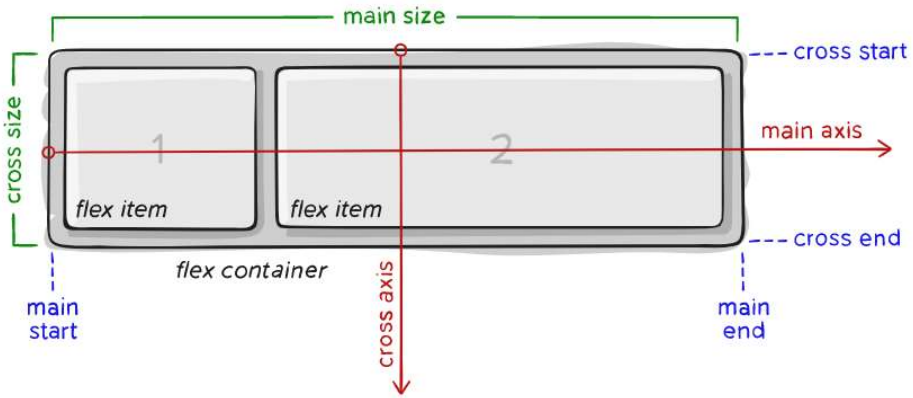
Flex Layout

KodeGo

cross axis

– The axis perpendicular to the main axis is called the cross axis.

- Its direction depends on the main axis direction.



cross-start | cross-end

– Flex lines are filled with items and placed into the container starting on the cross-start side of the flex container and going toward the cross-end side.

cross size

– The width or height of a flex item, whichever is in the cross dimension, is the item's cross size.

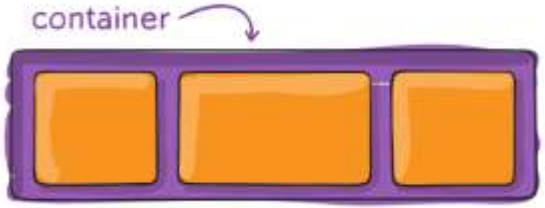
- The cross size property is whichever of 'width' or 'height' that is in the cross dimension.

FLEXBOX PROPERTIES

Properties for the Parent

(flex container)

- this is a flex container (the purple area) with three flex items:



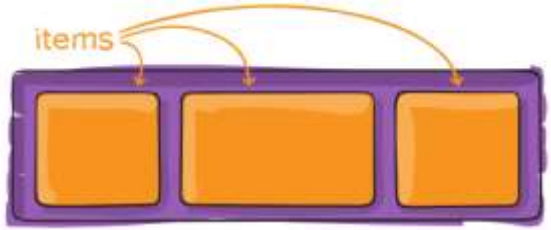
- The flex container becomes flexible by setting the display property to flex.

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

Properties for the Children

(flex items)

- The direct child elements of a flex container automatically becomes flexible (flex) items.



- The element above represents three orange flex items inside a purple flex container.

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

KodeGo

FLEX CONTAINER PROPERTIES

The flex container properties are:

1. flex-direction
2. flex-wrap
3. flex-flow
4. justify-content
5. align-items
6. align-content
7. gap, row-gap, column-gap

FLEX ITEM PROPERTIES

The flex item properties are:

1. order
2. flex-grow
3. flex-shrink
4. flex-basis
5. flex
6. align-self

FLEX CONTAINER PROPERTIES

KodeGo

display

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

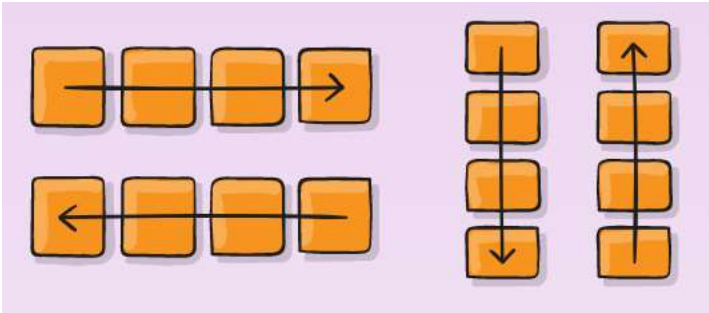
```css
.container {
  display: flex; /* or inline-flex */
}
```

FLEX CONTAINER PROPERTIES

KodeGo

1. 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.



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

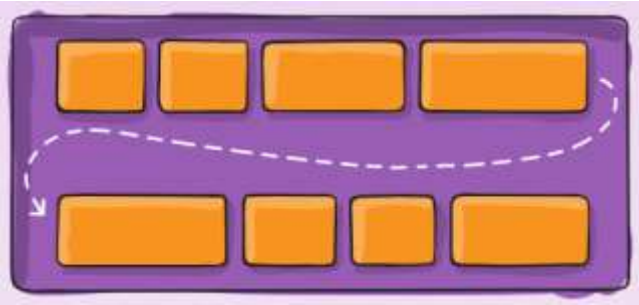
- row (default): left to right
- row-reverse: right to left
- column: same as row but top to bottom
- column-reverse: same as row-reverse but bottom to top

FLEX CONTAINER PROPERTIES

KodeGo

2. 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.
- wrap-reverse: flex items will wrap onto multiple lines from bottom to top.

FLEX CONTAINER PROPERTIES

KodeGo

3. 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 nowrap.

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

FLEX CONTAINER PROPERTIES

KodeGo

4. 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.

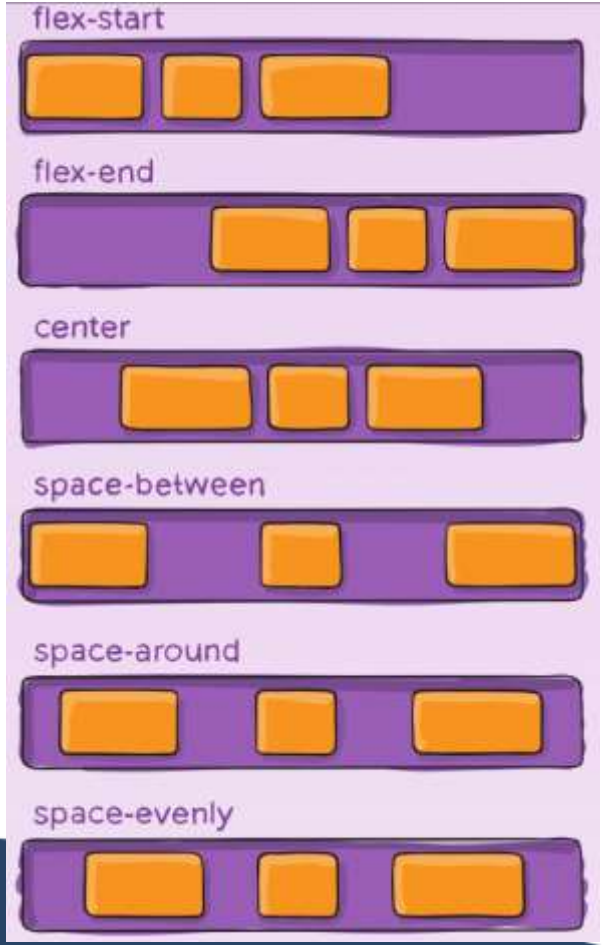
```
.container {
                justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right ... + safe | unsafe;
}
```

FLEX CONTAINER PROPERTIES

KodeGo

4. justify-content

- flex-start (default): items are aligns at the beginning of the container
- flex-end: items are aligns at the end of the container.
- center: items are aligns at the center of the container.
- space-between: items are displayed with space between the items.
- space-around: item are displayed with space before and between and after the items.
- space-evenly: items are displayed with equal space around them.



FLEX CONTAINER PROPERTIES

KodeGo

5. 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).

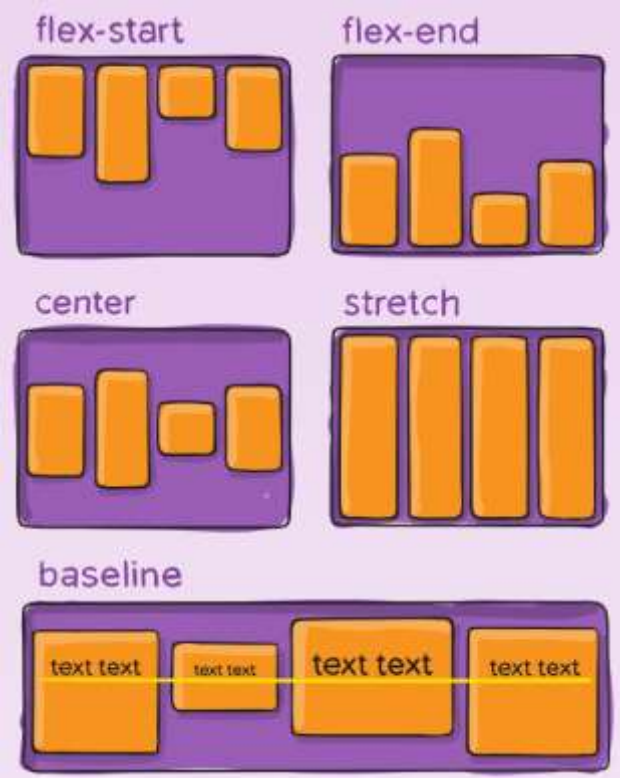
```
.container {
                align-items: stretch | flex-start | flex-end | center | baseline | first baseline | last baseline |
start | end | self-start | self-end + ... safe | unsafe;
}
```

FLEX CONTAINER PROPERTIES

KodeGo

5. align-items

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



FLEX CONTAINER PROPERTIES

KodeGo

6. 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.

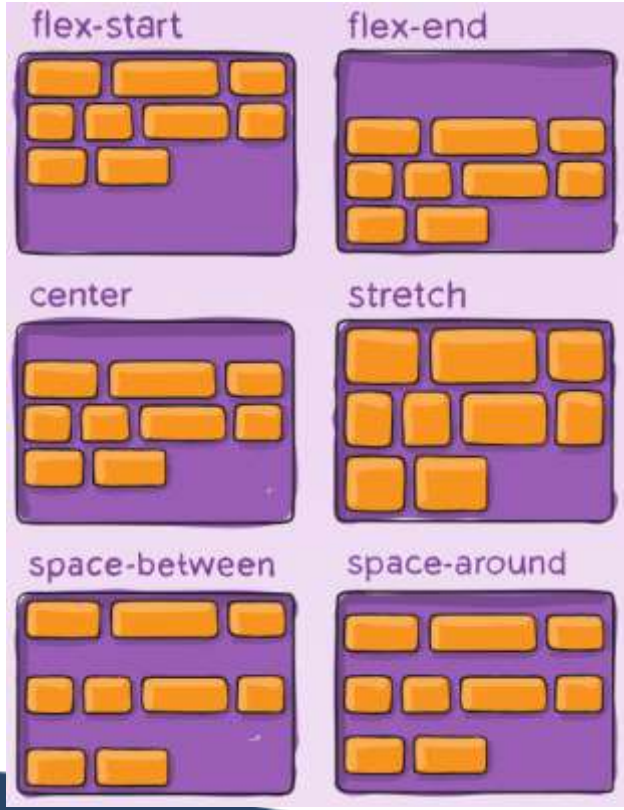
```
.container {
                align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline | last baseline + ... safe | unsafe;
}
```

FLEX CONTAINER PROPERTIES

KodeGo

6. align-content

- 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

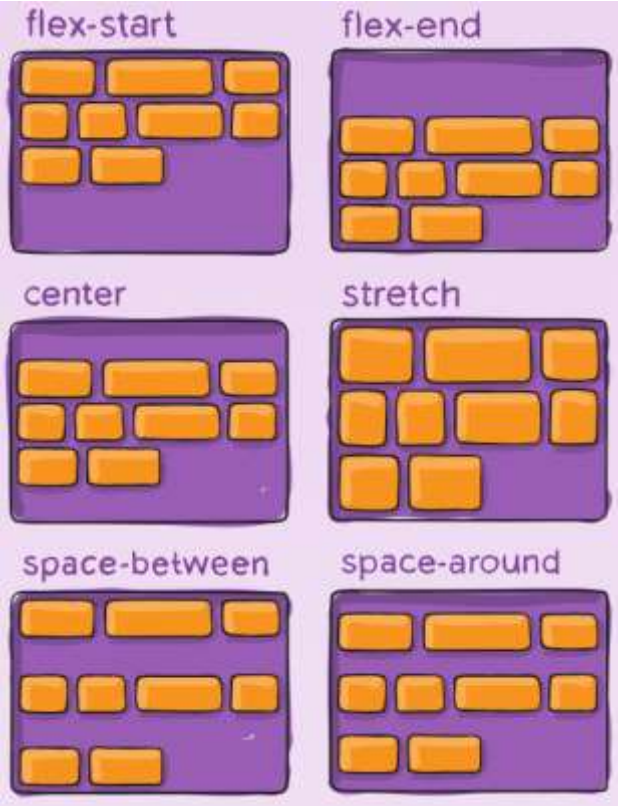


FLEX CONTAINER PROPERTIES

KodeGo

6. align-content

- 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



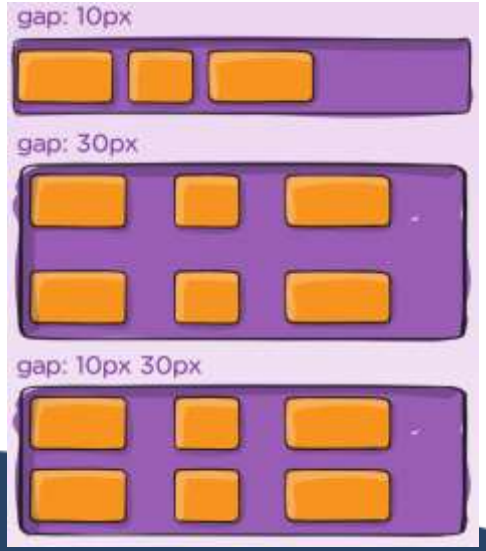
FLEX CONTAINER PROPERTIES

KodeGo

7. gap, row-gap, column-gap

- The gap property explicitly controls the space between flex items. It applies that spacing only between items not on the outer edges.

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

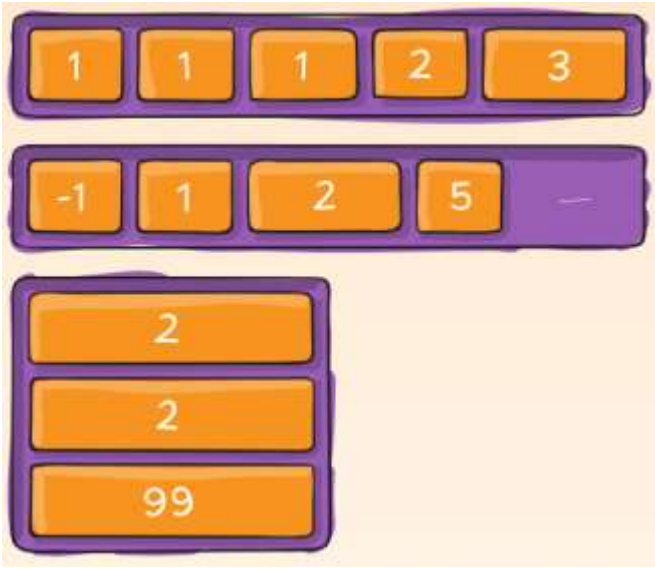


FLEX ITEM PROPERTIES

KodeGo

1. 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.
- Items with the same order revert to source order.



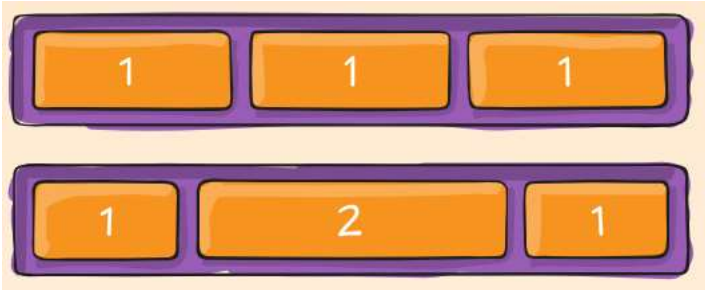

```
.item {
        order: 5; /* default is 0 */
}
```

FLEX ITEM PROPERTIES

KodeGo

2. flex-grow

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




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

FLEX ITEM PROPERTIES

KodeGo

3. flex-shrink

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

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

FLEX ITEM PROPERTIES

KodeGo

4. 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.

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

FLEX ITEM PROPERTIES

KodeGo

5. 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.
- The default is 0 1 auto, but if you set it with a single number value, like flex: 5;, that changes the flex-basis to 0%, so it's like setting flex-grow: 5; flex-shrink: 1; flex-basis: 0%;.

```
.item {
                flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

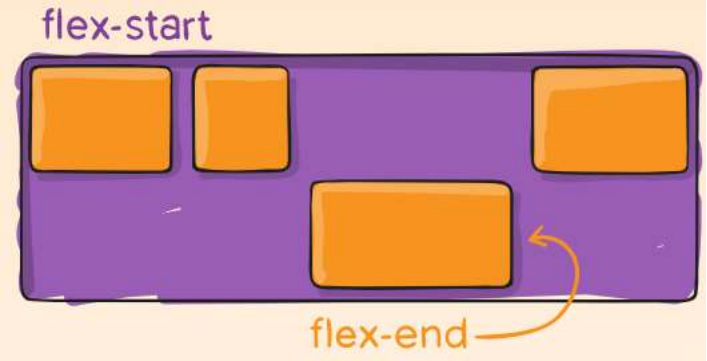
FLEX ITEM PROPERTIES

KodeGo

6. 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.




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

BROWSER SUPPORT

- A number indicates that browser supports the feature at that version and up.

Exercise

KodeGo