

When we do

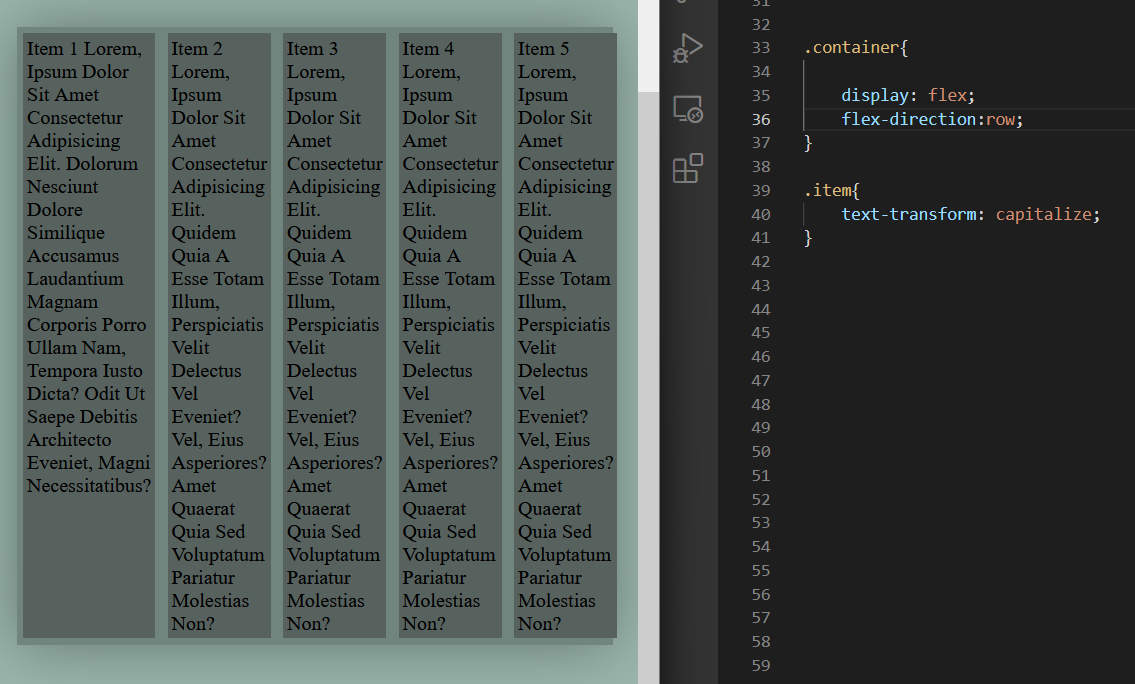
display:flex;

then the default one is

flex-direction:row; even if we do not mention it

here main axis will be along the row, and items always follows the main axis

See below



As seen in the first column the height is no longer auto, although being block level element. Reason see below

flex-direction:column;

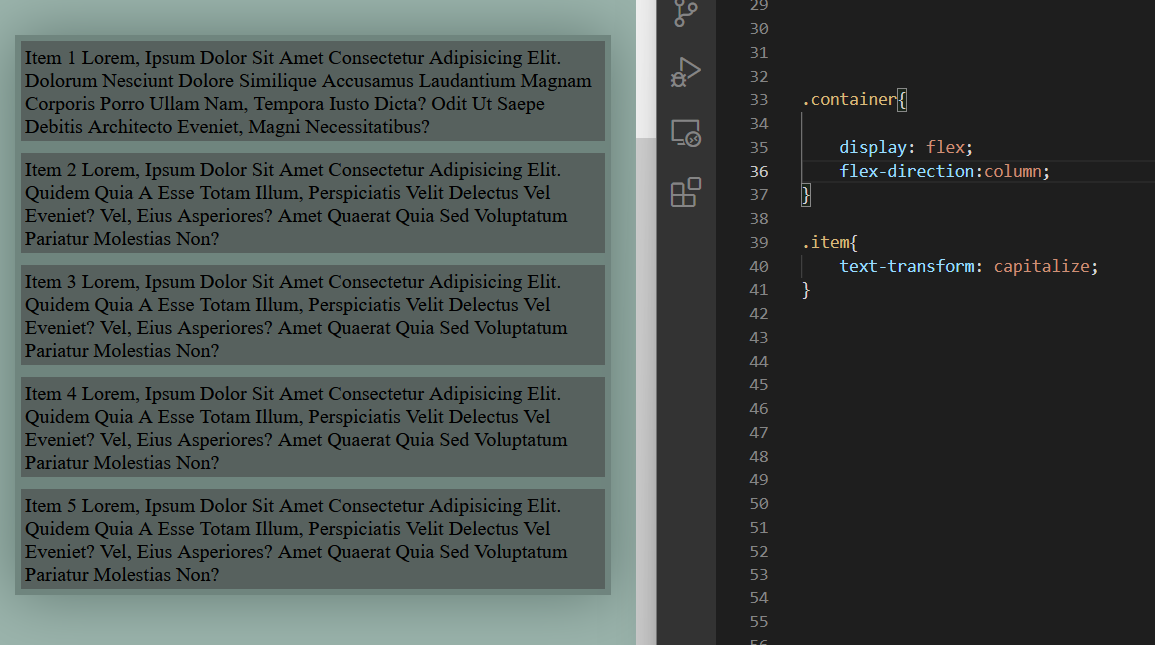
here items are arranged as columns inside container

or

Understand this the other way / the simpler way

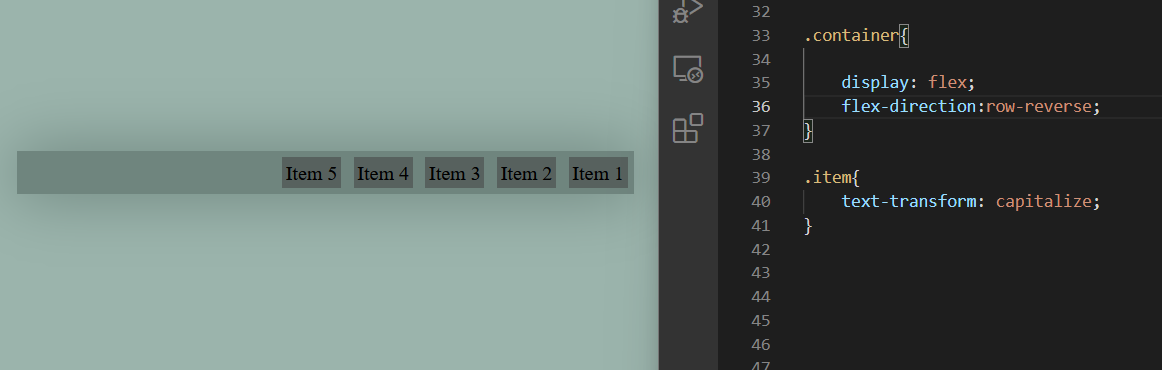
* doing flex-direction:column; main axis will be along the column
* and the items always follows the main axis

here the fact that each items is taking whole width is being the block level element



flex-direction:row-reverse;

here items still arranges along the row, but from reverse



when we do

flex-direction:row;

The **flex-direction** [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) property sets how flex items are placed in the flex container

row meaning items are arranged as row inside the container

* the main axis will be along row
* and the items always follows the main axis

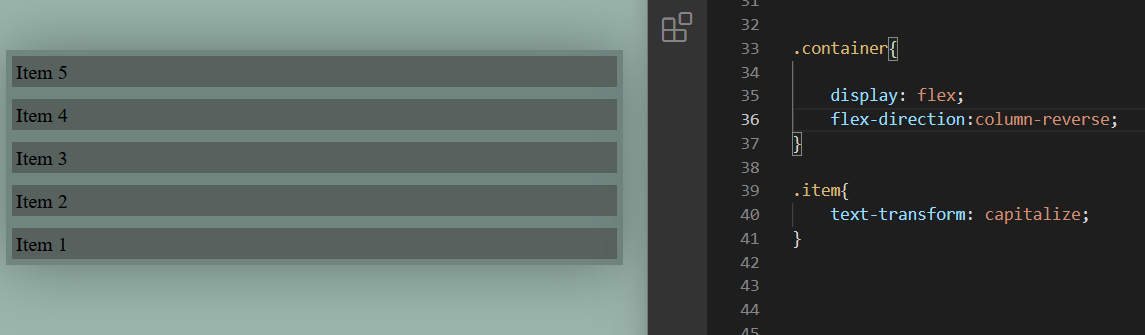
imagine an arrow on left side of container and from that side container becomes a row

so on doing row-reverse

the container becomes a row from the backward

and items arrange themselves from there

flex-direction:column-reverse;



imagine an arrow

in column reverse imagine that arrow at the bottom of container box

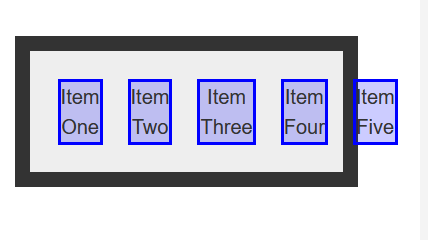
so container becomes column from bottom

so items arrange themselves in that way

flex-wrap:

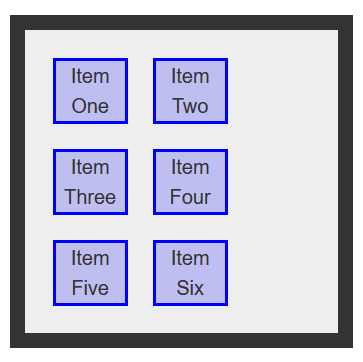
The **flex-wrap** [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) property sets whether flex items are forced onto one line or can wrap onto multiple lines

flex-wrap:nowrap;

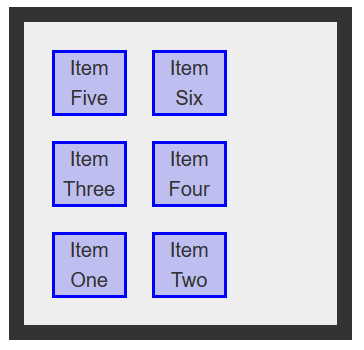


it is the by default property of flexbox

flex-wrap:wrap;



flex-wrap:wrap-reverse;



Also when the items are wrapping, they consider the flex direction

if

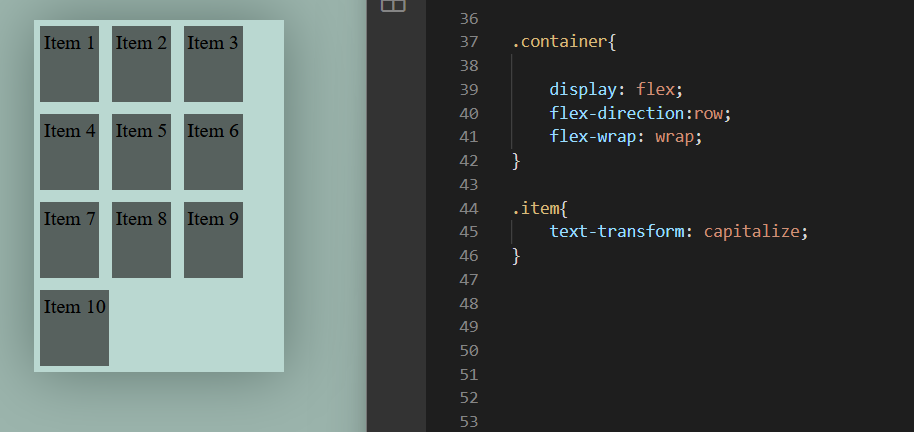
flex-direction:row;

than items wrap row wise

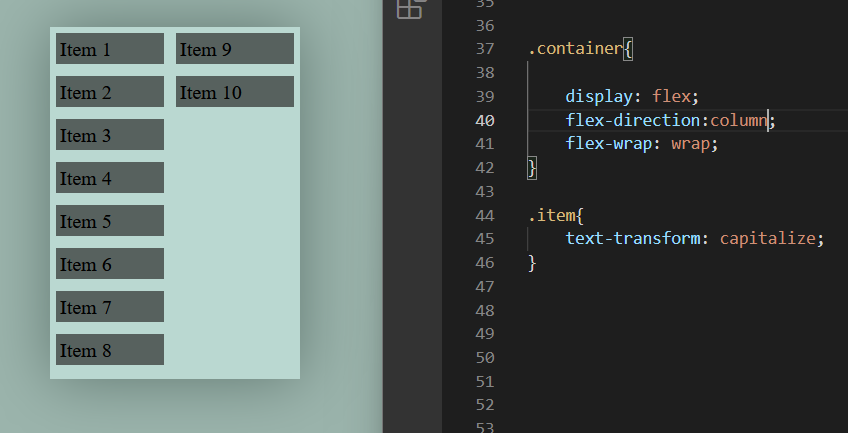
if

flex-direction:column;

items wrap column wise



items are wrapping inside container row wise



items are wrapping inside container columns wise

flex-flow;row wrap;

flex-flow:column nowrap;

it’s just a short hand property for flex direction and flex wrap

**Better than flex-direction**

Instead of flex direction we can use

flex-flow

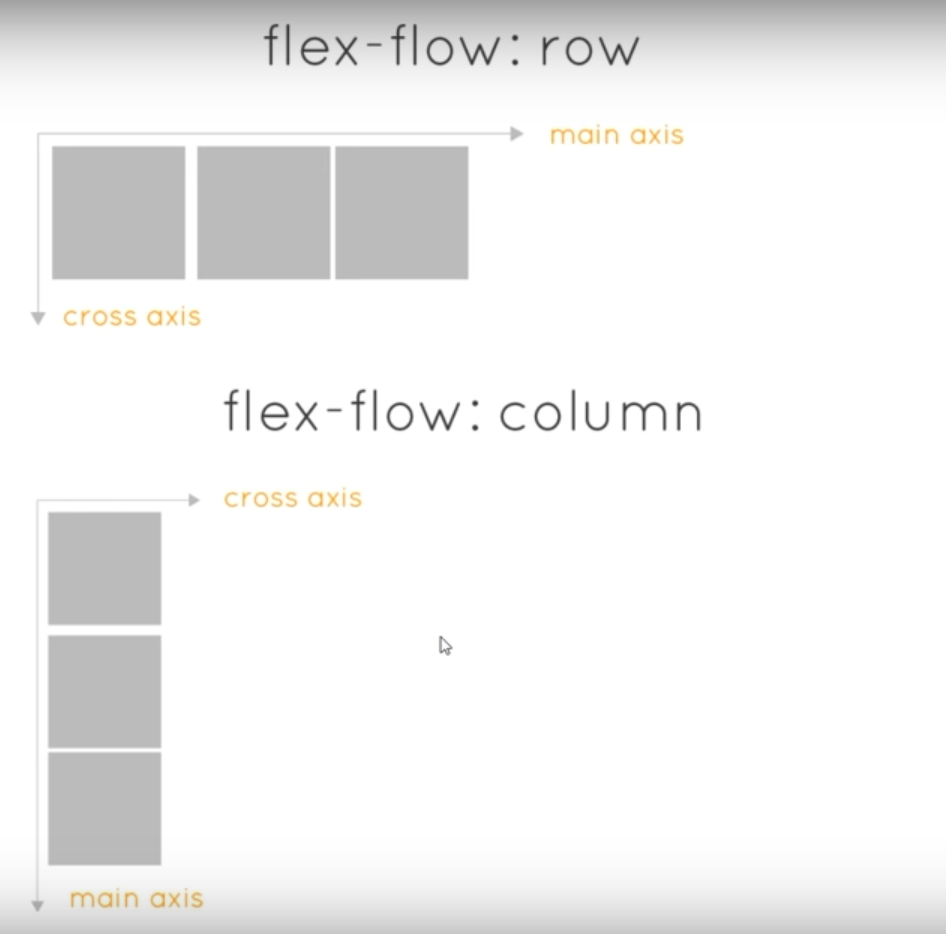
flex-flow:row;

meaning we want our flex items to flow as rows

flex-flow:columns;

meaning we want our items to flow like columns

and there are also row-reverse and column-reverse



**Now**

**There are certain properties that only applies to the main axis**

**and**

**There are certain properties that only applies to the cross axis**

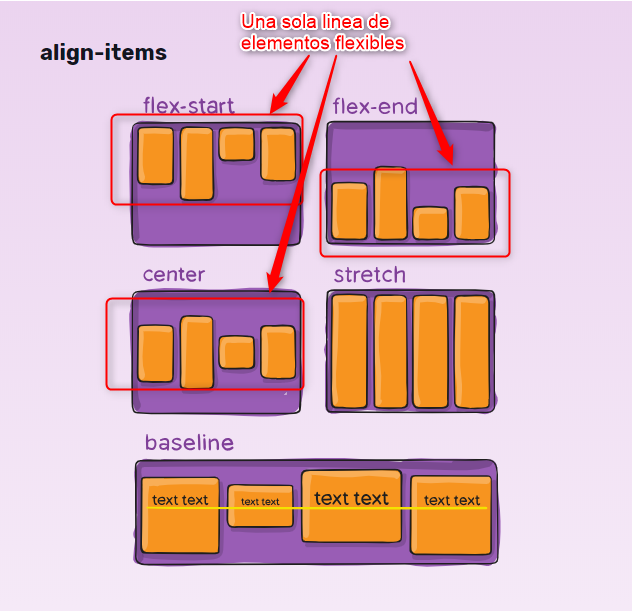
justify-content:

It applies along the main axis

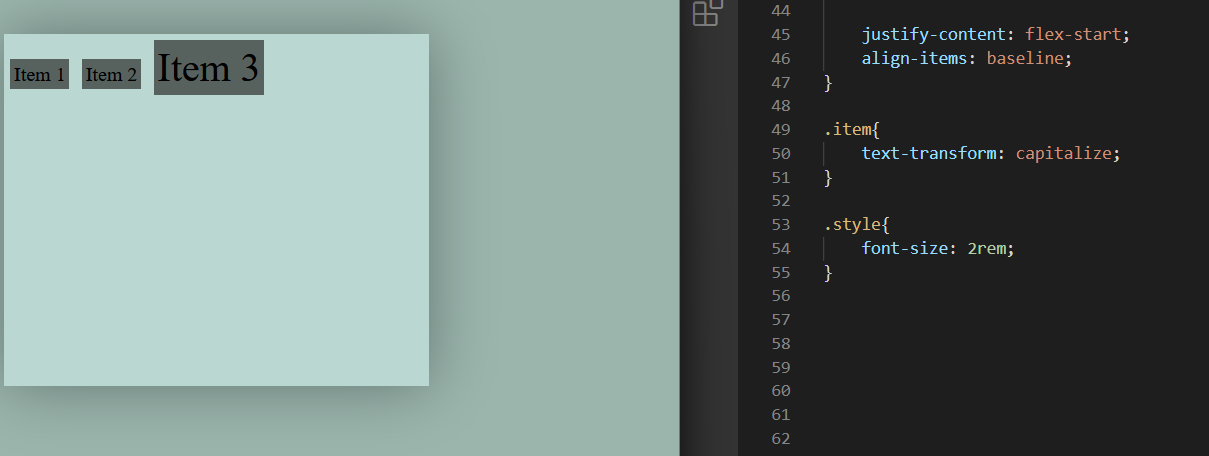
align-items:

It applies along our cross axis

and our main axis are different for flex-flow: row or column;

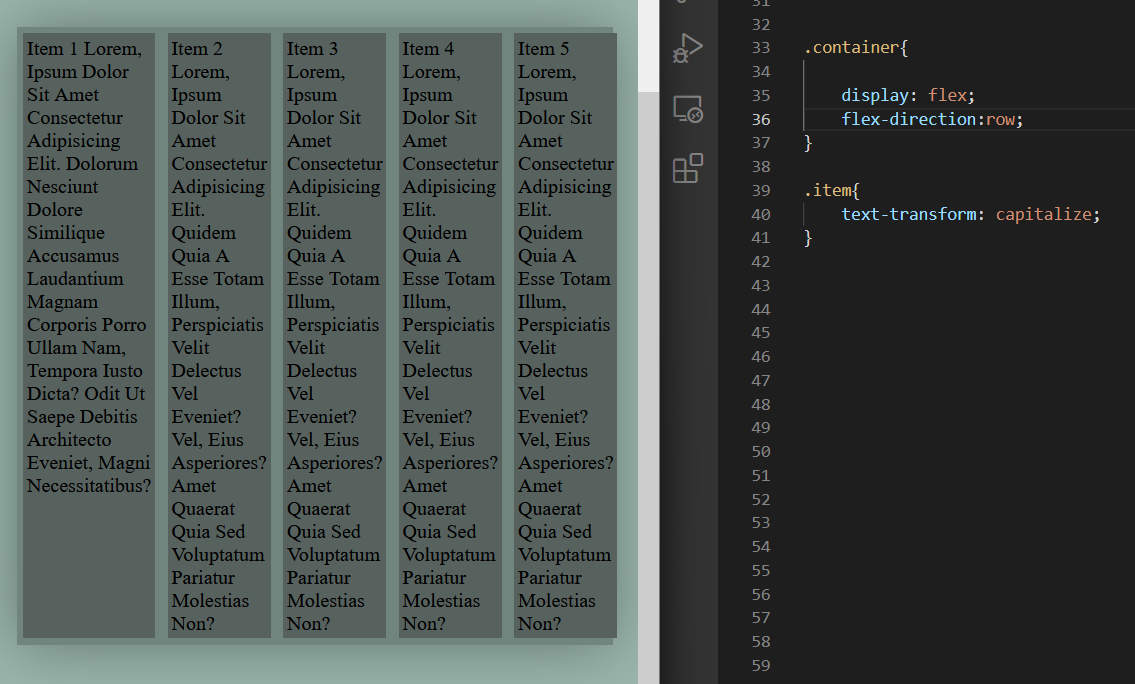


The baseline is a term used in European and West Asian **typography meaning an imaginary line upon which the characters of a font rest**. The descenders of characters like g and p extend below this line



this

align-items:baseline; can be really handy at various situations



This is weird

In css

By default

Block level element like<h1> <h2> <div>

always occupy whole width along the horizontal direction, but the height is not whole along the vertical direction

But the height is always auto, meaning the height is based on the content

The height is just to fit the content

The height varies, grows and shrinks as per the content

But above in first item there is little gap after the content ends,

which isn’t case of height:auto; for the block level element <div>

**Reason**

By default

align-items:stretch; property this happened

in css flexbox

so this happened

should i use flexbox or grid for layout ?

ans: CSS grids are for 2D layouts. It works with both rows and columns. **Flexbox works better in one dimension only** (either rows OR columns). It will be more time saving and helpful if you use both at the same time.

**Leaving align content and justify items for now**

flex-grow:

It gives the growing factor to each items

see





here, when

container width is largest the flex grow works

when

container width is smaller

the flex grow doesn’t works

the items are maintained at minimum width by min-width or mostly flex-basis: property

*Understand this as : We have set minimum width and how the items grows as per container width increases, and we can set flex growth rate for each item*

flex-shrink :

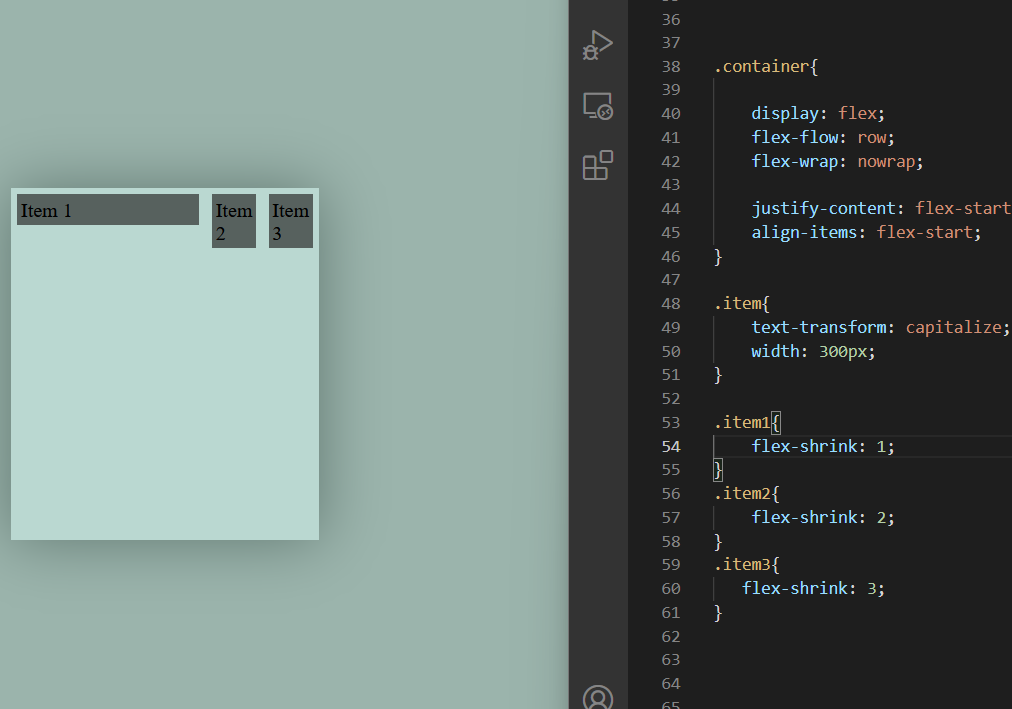
we set the width for the flex items and as browser gets smaller we can choose the rate by which each items shrinks.

But

flex-wrap

property of the flex container may contradict with flex shrink property

so see carefully



we set width 300 px

so when browser gets smaller the items shrinks as per the provided values

also

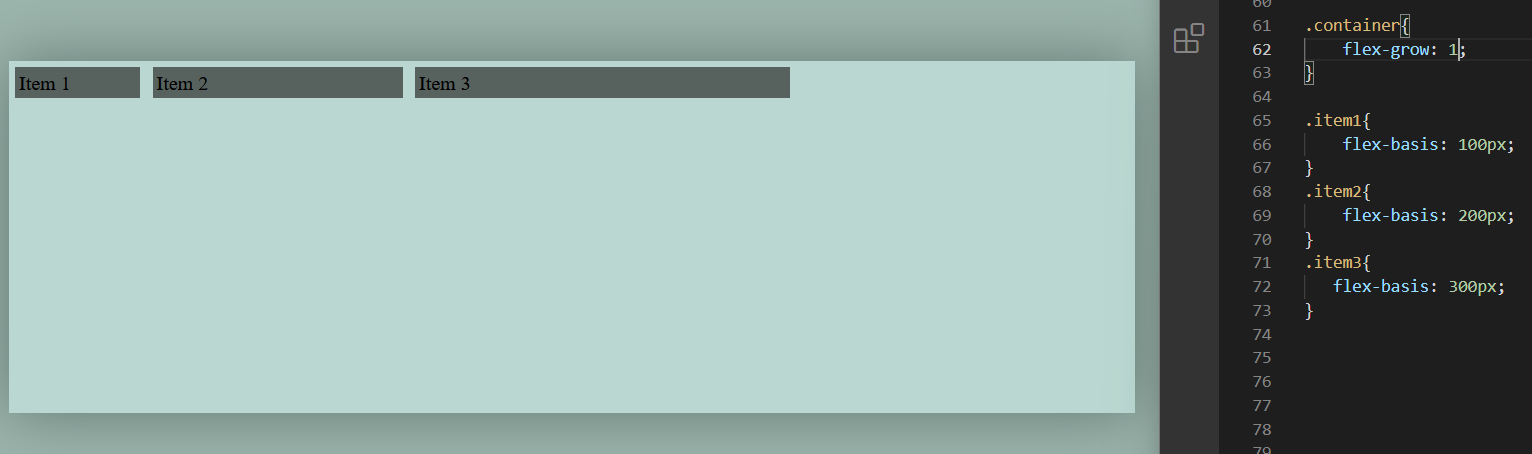
see

we set

flex-wrap:nowrap;

else it won’t work

**flex-basis and flex-grow**



first item width :100 px;

second item : 200px;

third item: 300px;

flex-basis, this sets the width of the items, and now

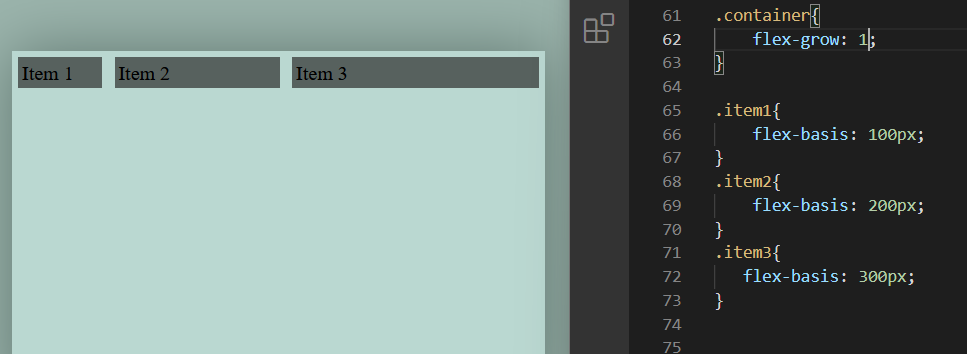
flex-grow:1;

is applied on the container

so each item grows uniformly as we

increase the size of the browser

but due to varying flex-basis size it may not seem uniform



here by flex basis the base size of each item was 100,200 and 300 px

but these items will grow above flex basis and shrink above flex basis

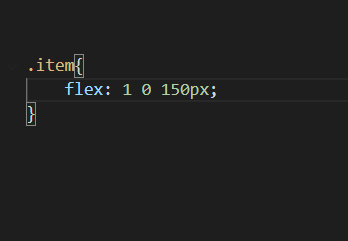
**We use short hand property, consisting of all flex-basis, flex-grow and flex-shrink**

flex : 1 0 100px;

1st  : flex grow

2nd : flex shrink

3rd : flex-basis





here the flex-grow is 1 for all item, so all grows uniformly

flex basis is 150px,

so items grows uniformly from size 150 pixels

but since flex-shrink is 0, so items won’t shrink from 150 px

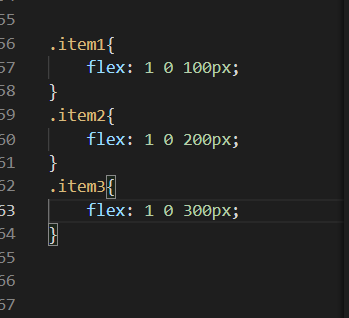
see below

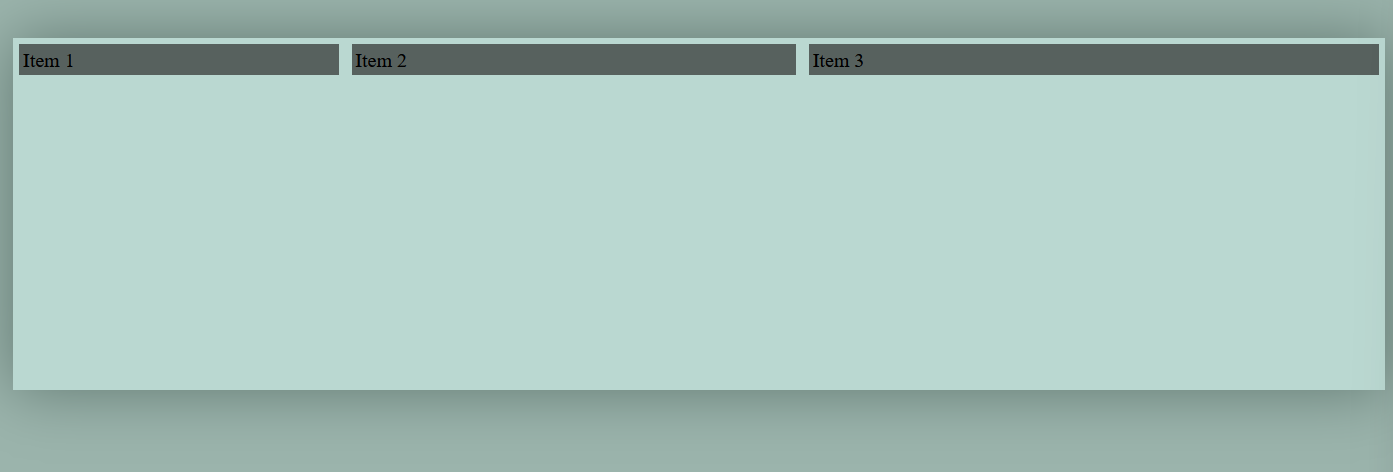


see items are not shrinking in to the container

but if we set

flex:1 1 150px; than items will also shrink

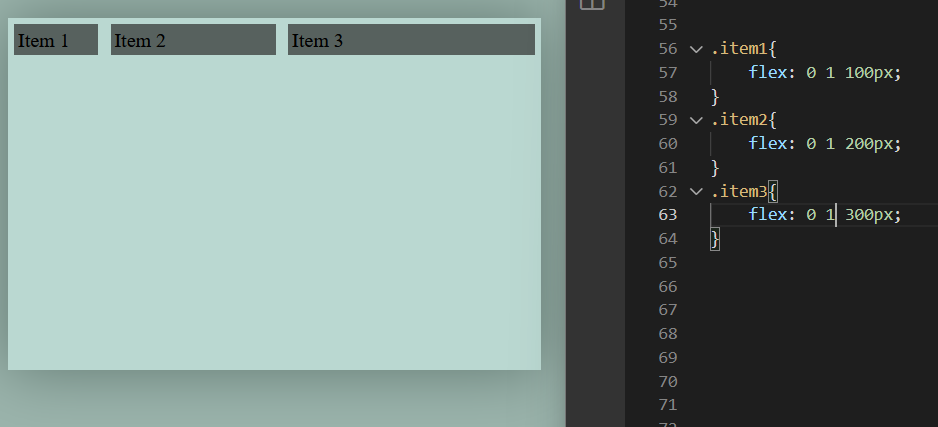




here each item is provided different flex basis size

and flex shrink is 0 for all so items can not shrink in to the container when we decrease browser’s size

also flex grow is 1, so all grows uniformly above flex basis size



unlike above here is flex grow 0, so items won’t grow above flex basis

and items will shrink below flex basis

**Very handy flex-box property**

well if we want to reverse the order of all the flex items

than we have

flex-direction:row-reverse;

and

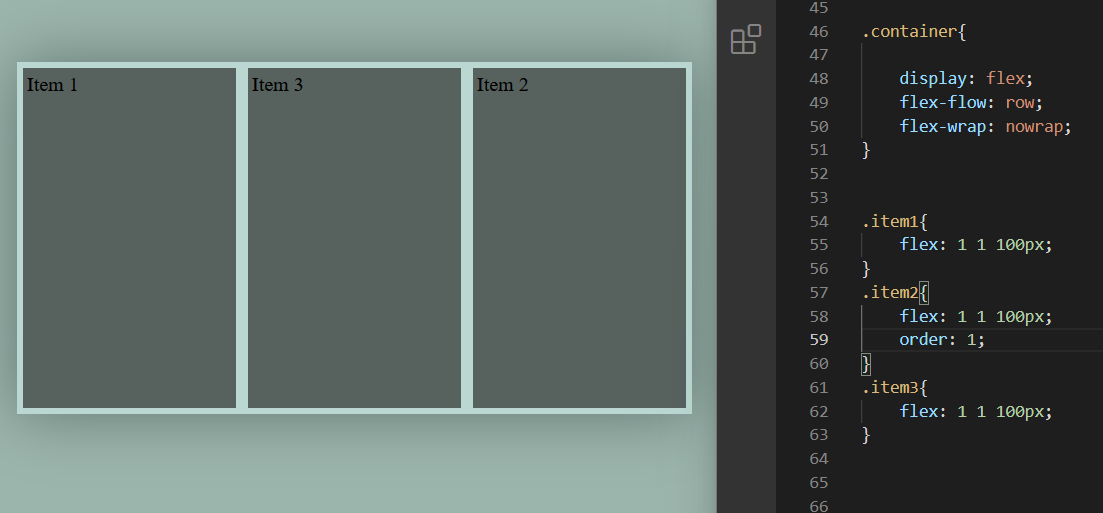
flex-direction:column-reverse;

but when we just want to took one flex item to last from rest of all

we can do that from

order: property

the item having highest order goes at last



rule:

by default

order:0;

here item1 and item3

have by default order:0;

but item2 has order:1;

item2 has more order than rest

so it goes to the last

the item having lowest order goes to the front / and we can also set the negative values

