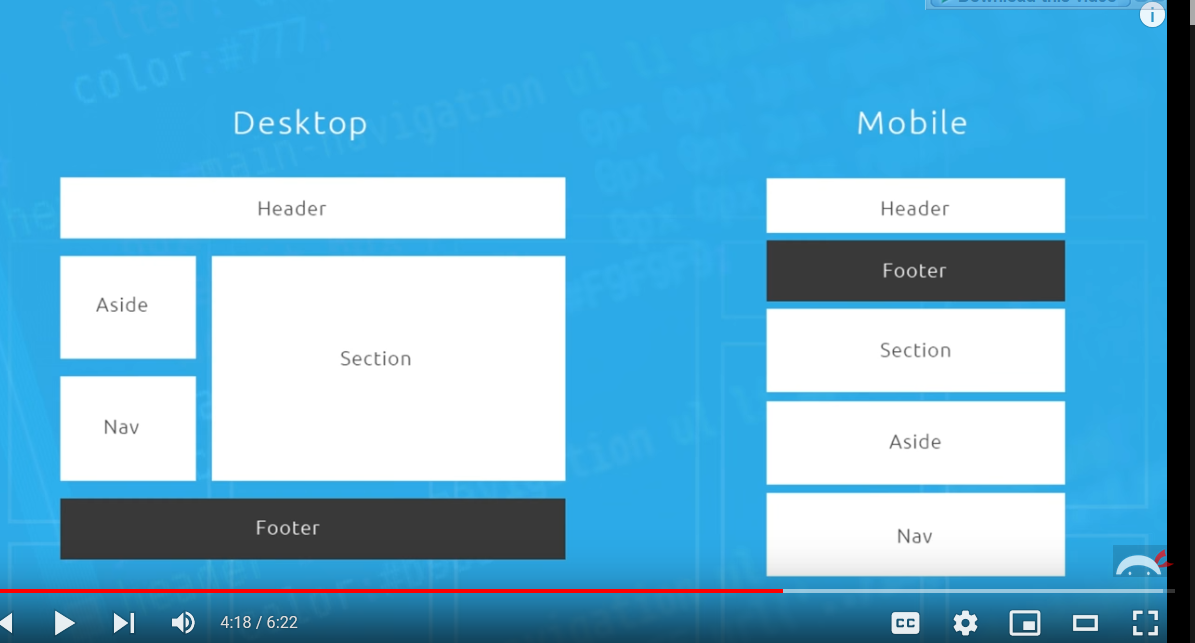
**Mostly :**

**All about CSS Grid, think of css Grid a tool to layout whole site, where as flexbox to layout sections in that larger layout**



we can design any type of layout for any device type by grid

Also

see above, in mobile device

the footer is at second

well actually

In grid we can play around with elements like this

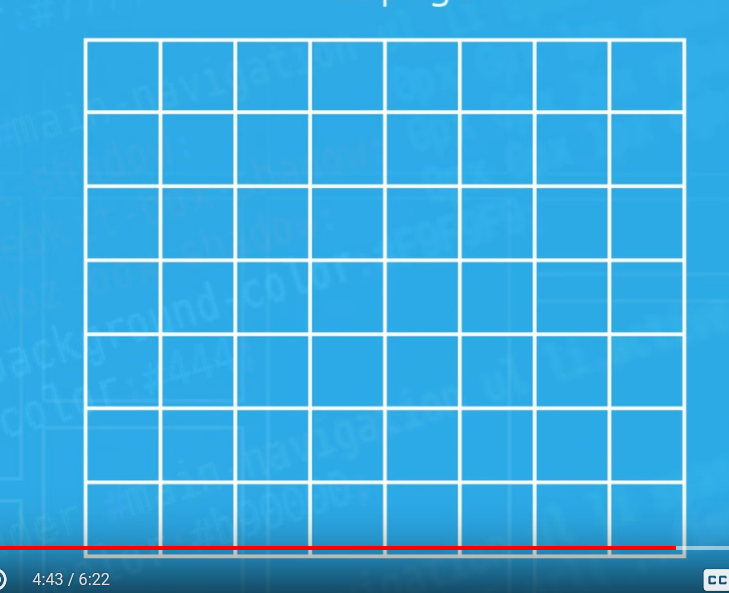
it doesnot care about order

In flexbox there was an property

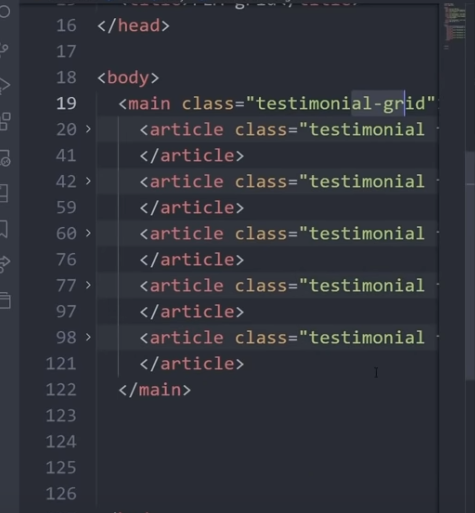
order:

by which also we could play around elements

but we can do same here in grid also



It treats the web page as a grid, where we can define the grid lines and we make our layout in that grid



seems like such html layout is easier to style in grid

In grid avoid using margin

start using gap

gap:1.2rem;

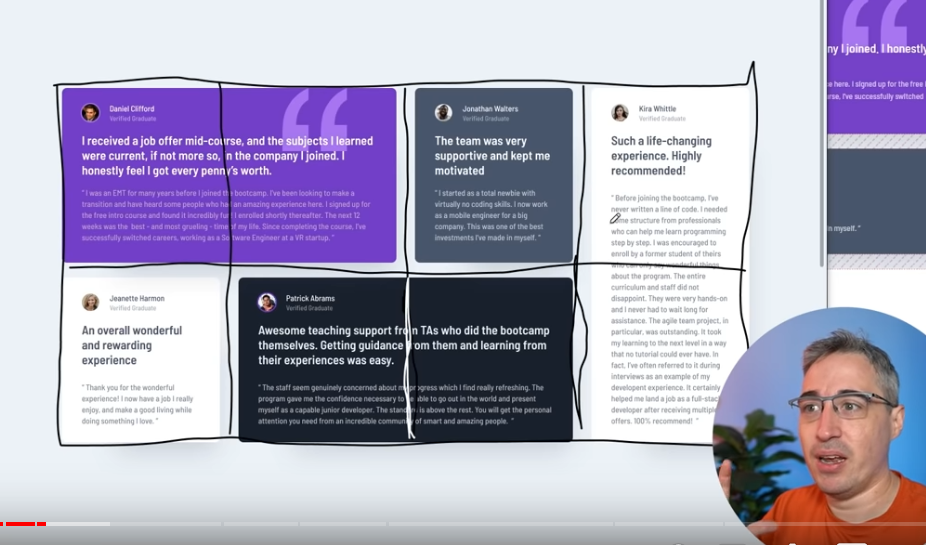
Advantage over margin

gap only creates gap inside the grid, meaning it doesnot create any margin before 1st grid line and after last grid line

**One of the hardest thing is to determine how may columns we may need for our layout ?**

well try by drawing the grid lines on the layout that you are gonna build

like this



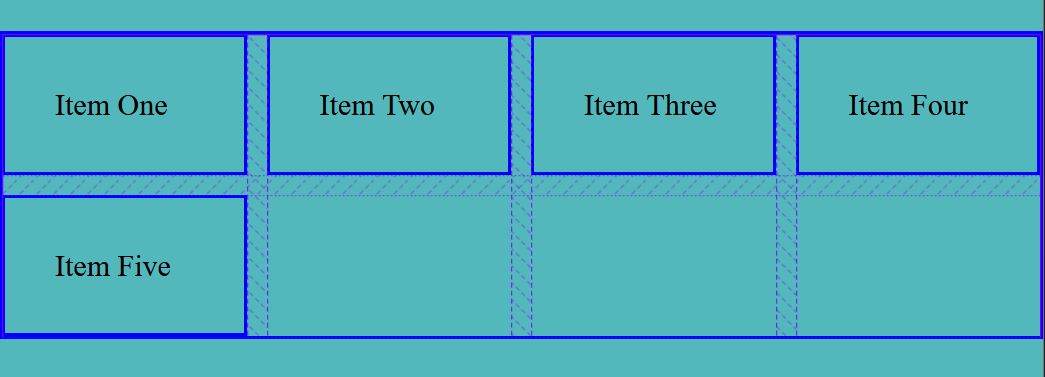
we see that clearly we need

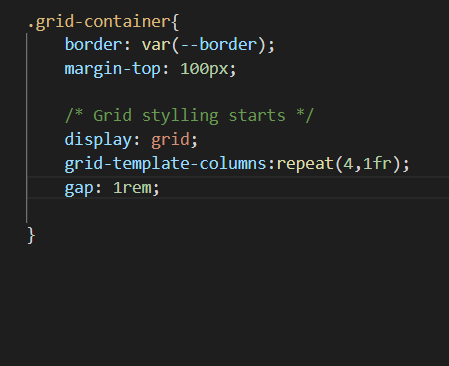
4 columns

and

two rows

for this screen size





As we can see the

gap:1rem;

gap do not set up margin at right and left of the grid-container

like margin does

also repeat(4,1fr)

fr : is a fractional unit specific only to grid container

and when we had to make columns of uniform width like

grid-template-column:1fr 1fr 1fr 1fr; ( it creates 4 columns each of equal width inside grid )

instead of above we do

grid-template-column:repeat(4,1fr)

but

doing

grid-template-column:1fr 2fr 1fr 2fr;

will give us columns of varying size

**When we make grid columns grid rows are made implicitely also**

Look closely above there we said

grid-template-column:repeat(4,1fr);

we don’t said anything about rows,

but since there is content to live in row

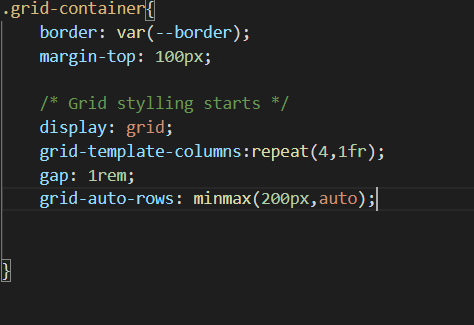
rows are implicitely made for us

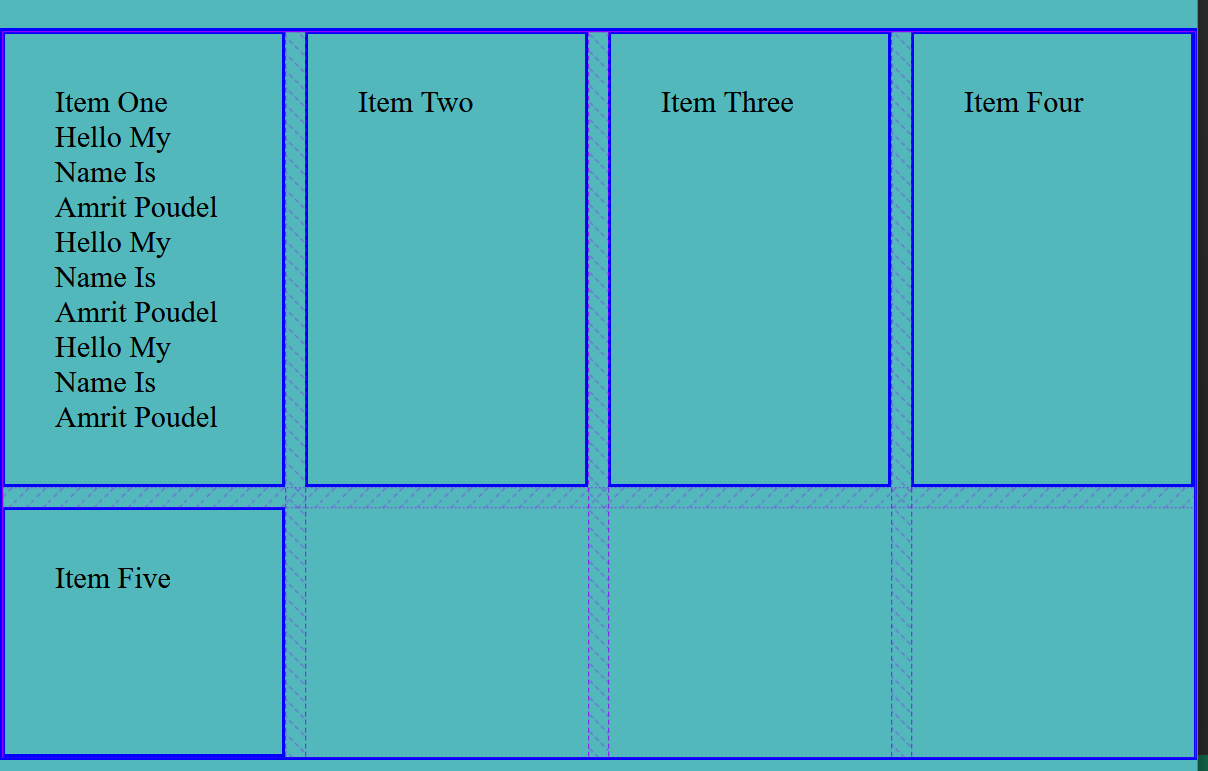
**Except in few specific case, we mostly declare grid columns and rows are implicitely made for us**

And also the height of row is determined by the height of highest item

row height = height of highest item

this may be annoying sometimes





we could also do

**grid-auto=rows: 200px or minmax(200px,auto) // meaning minimum height of 200px and maximum height is auto // and auto means the height of row is height of tallest item in container**

**The grid-auto-rows CSS property specifies the size of an auto-created grid row**

For size of explicitely created rows we go

grid-template-rows:repeat(4,minmax(200px,auto));

grid-auto-rows:200px; // this will make rows height 200px

by default the rows takes height of tallest item

grid-auto-rows:minmax(200px,1fr); // takes minimum height of 200px and maximum of 1fr, based on the content height in it’s item

In very specific case, we can also create rows of desired height like this

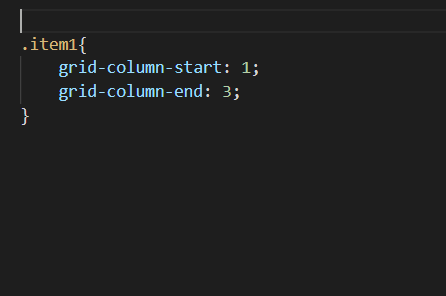
grid-template-rows: 120px 250px 300px 500px ;

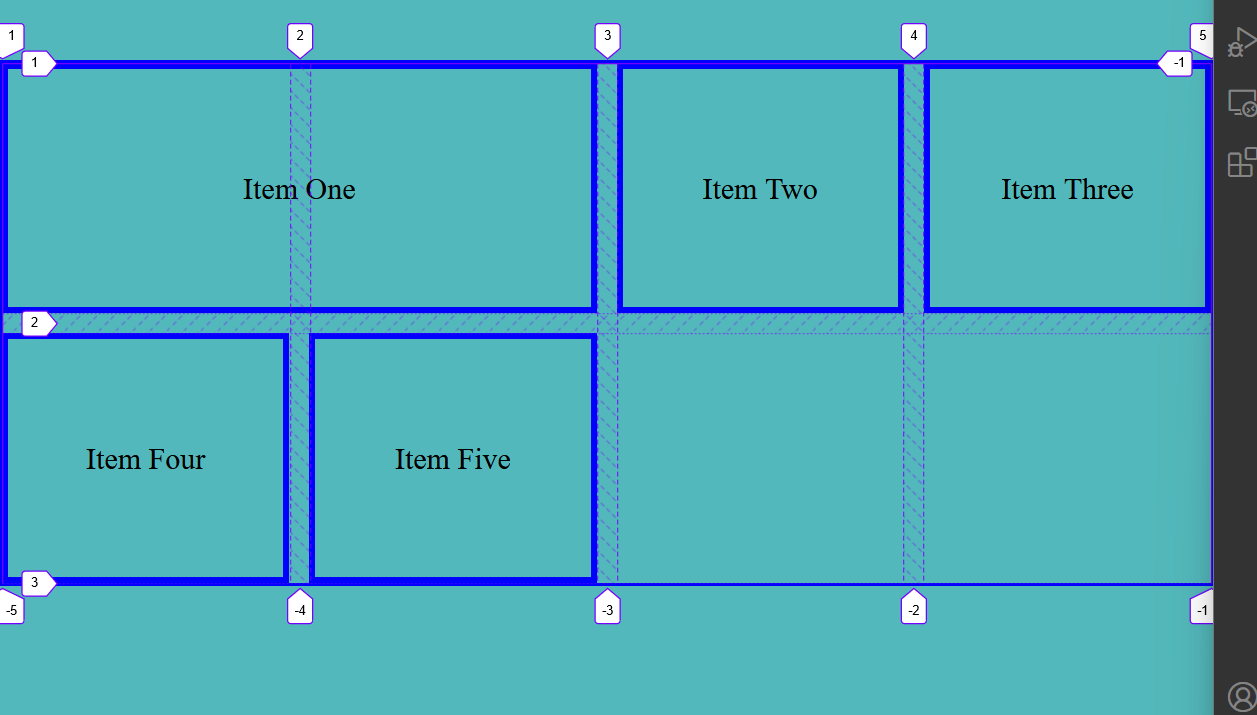
repeat( minmax( 250px,auto ) ) // sometimes we also use this

row-gap:

column-gap:

gap: it is a shorthand property for row-gap and column-gap





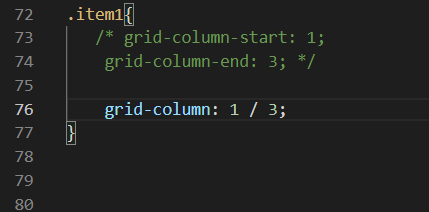
first of all turn on the grid lines from the firefox developer tools

grid-column-start : place the start column line

grid-column-end: place the end column line

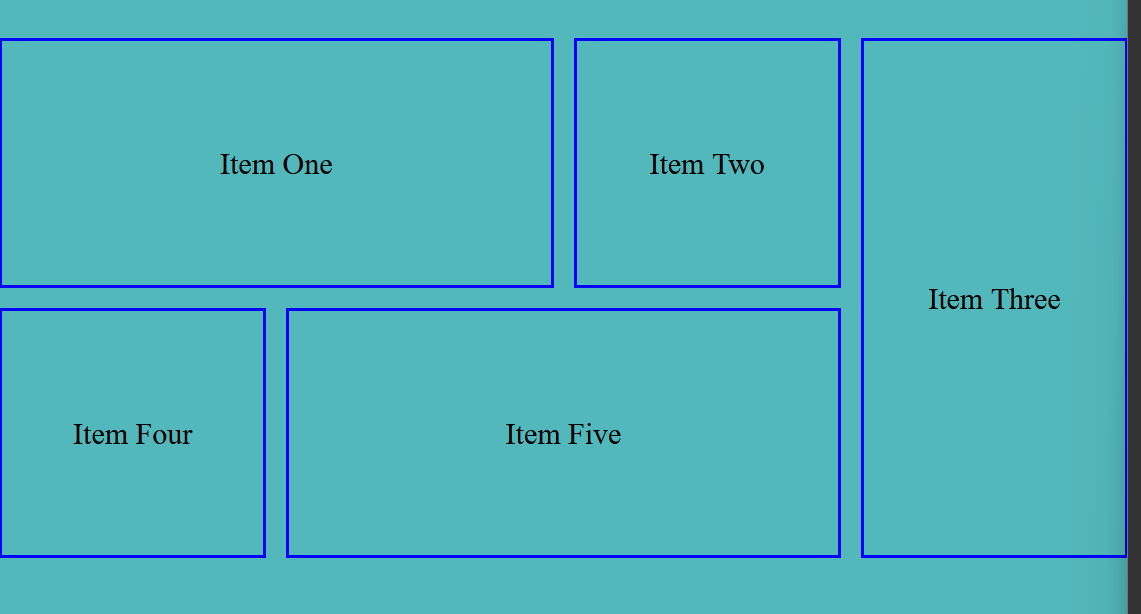
There is a shorthand property for this, but unlike others here we have /

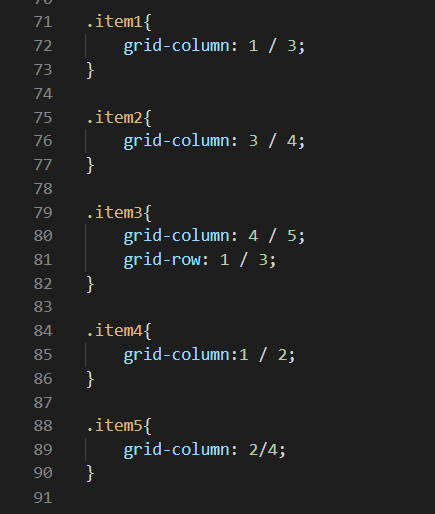
grid-column: 1 / 3;



Same for grid-row-start and grid-row-end and short hand property

grid-row:1 / 3;

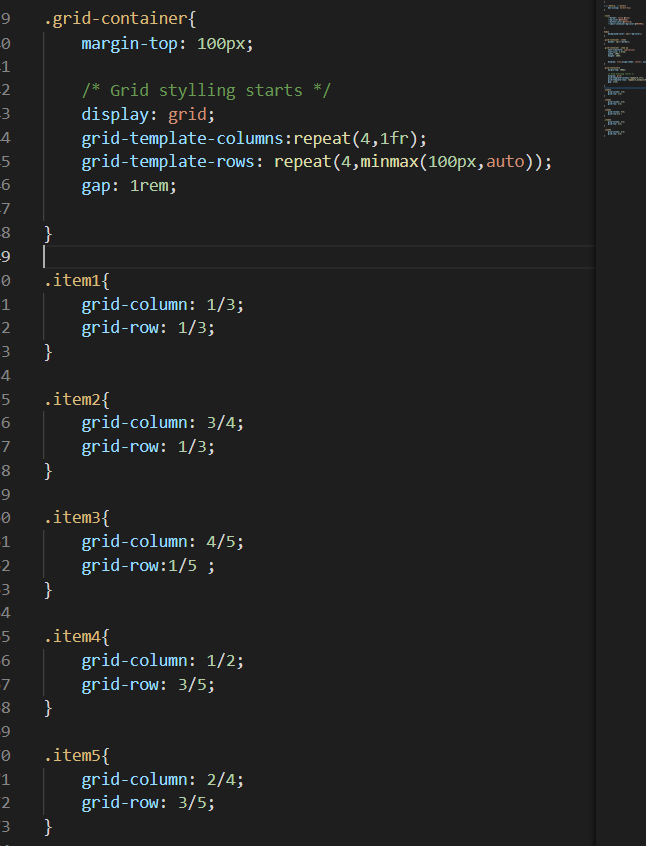


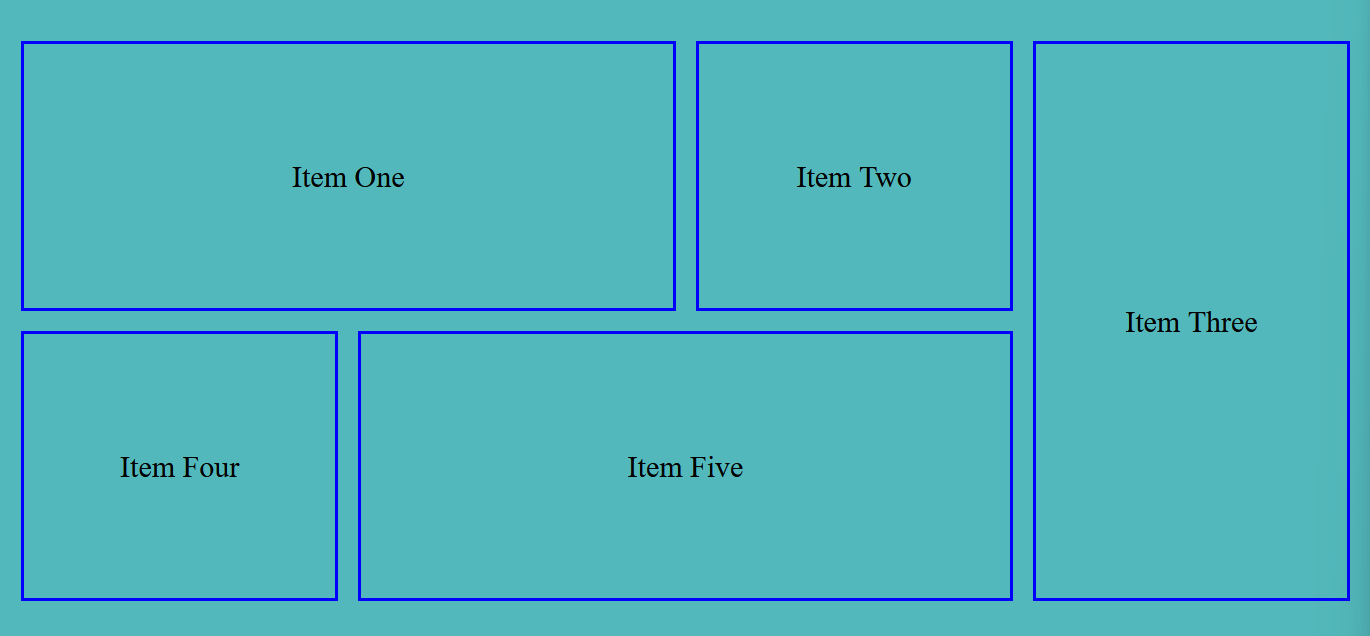


These shorthand-properties are just great

**Consider few things**

1. Here each item has min-width and min-height of 200px, and css grid is responsive, so each item section can be as large as grid container, but they can’t be smaller than min width and height
2. We mostly make columns, and let rows to be auto generated, except in specific case
3. Also when we donot make rows specifically the rows are auto created, and height of each created row is height of the tallest item
4. Also instead of maintaining each item of minimum width and height, we could make rows of minimum width and than we can make our items on them, now here we will be specific defining rows see below

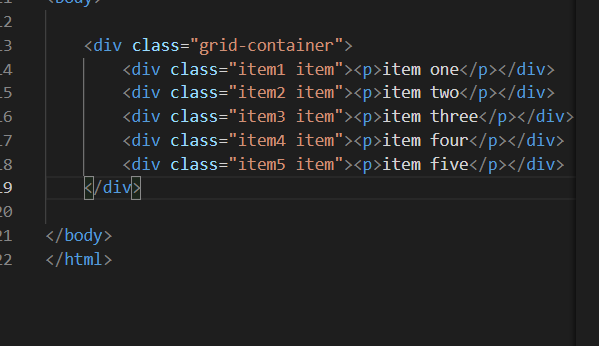


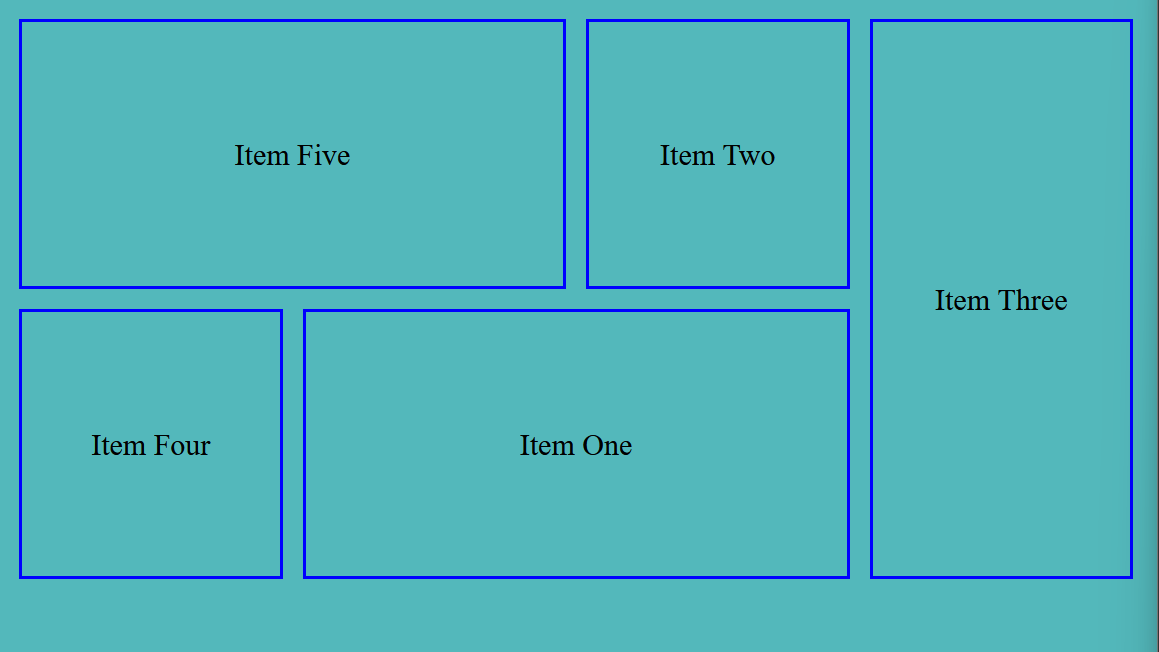


here each rows are made minimum of 100px, and we span our items among those rows

instead of specifically defining sizes of items

**CSS Grid doesnot care about the order, we can arrange items however we like in the grid**

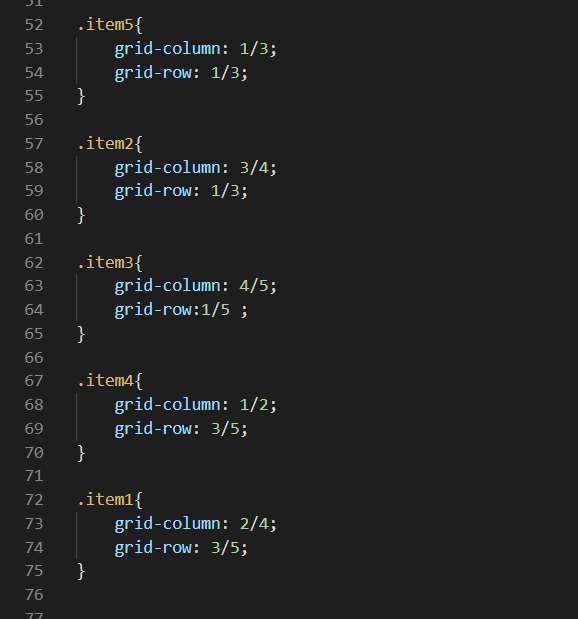
****



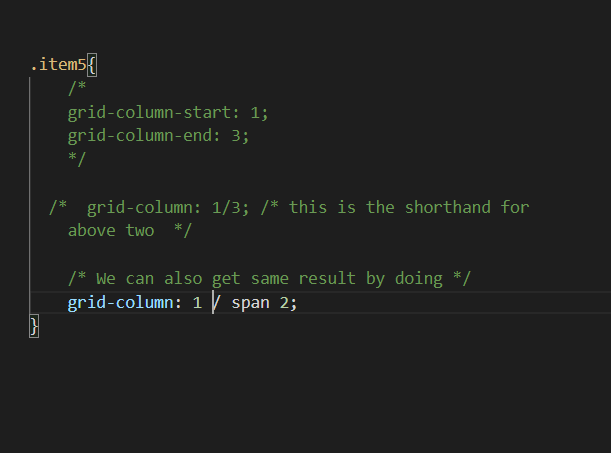
Unlike in html order, here item 5 is places at top left

Usually we may think only item 1 could be arranged like this

**This is the powerful thing provided by grid**



**We can also create nested grid, whenever we require**



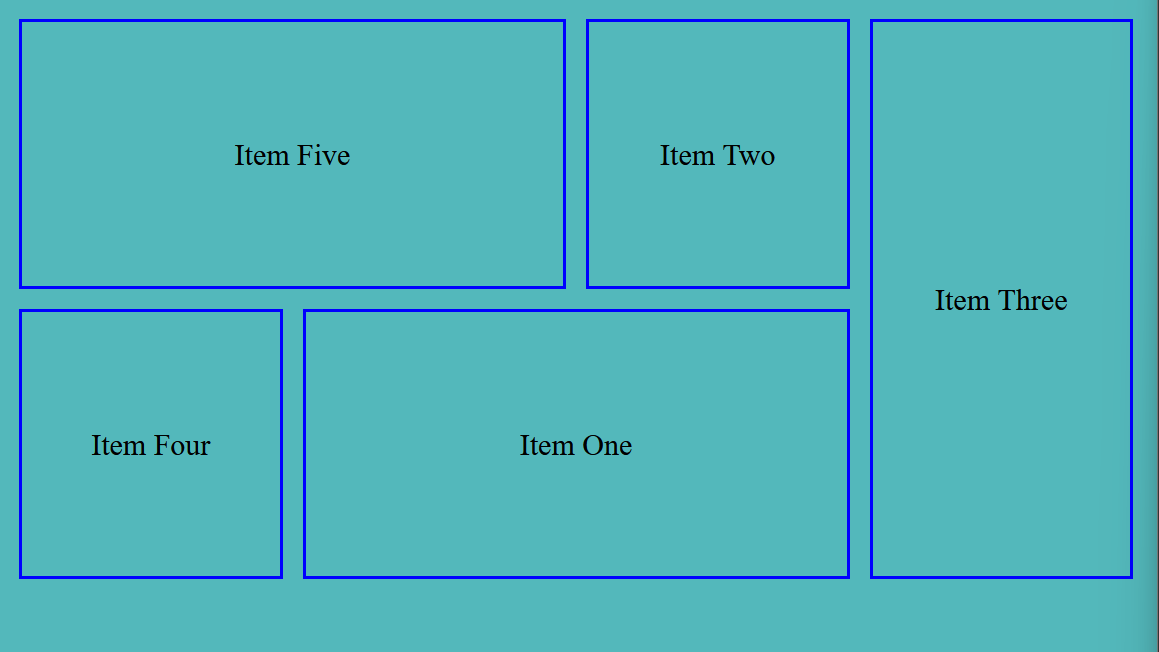
Item 5 start from column line 1 and then it span 2 columns

**justify-items and align-items in grid**

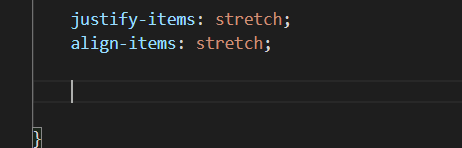
**here justify-items applies along main-axis i.e along x-axis i.e right, left**

**here align-items applies along the cross axis i.e along y-axis i.e top, bottom**

**Also these two properties applies in te grid container not to each items, so applies to all the items in that container**

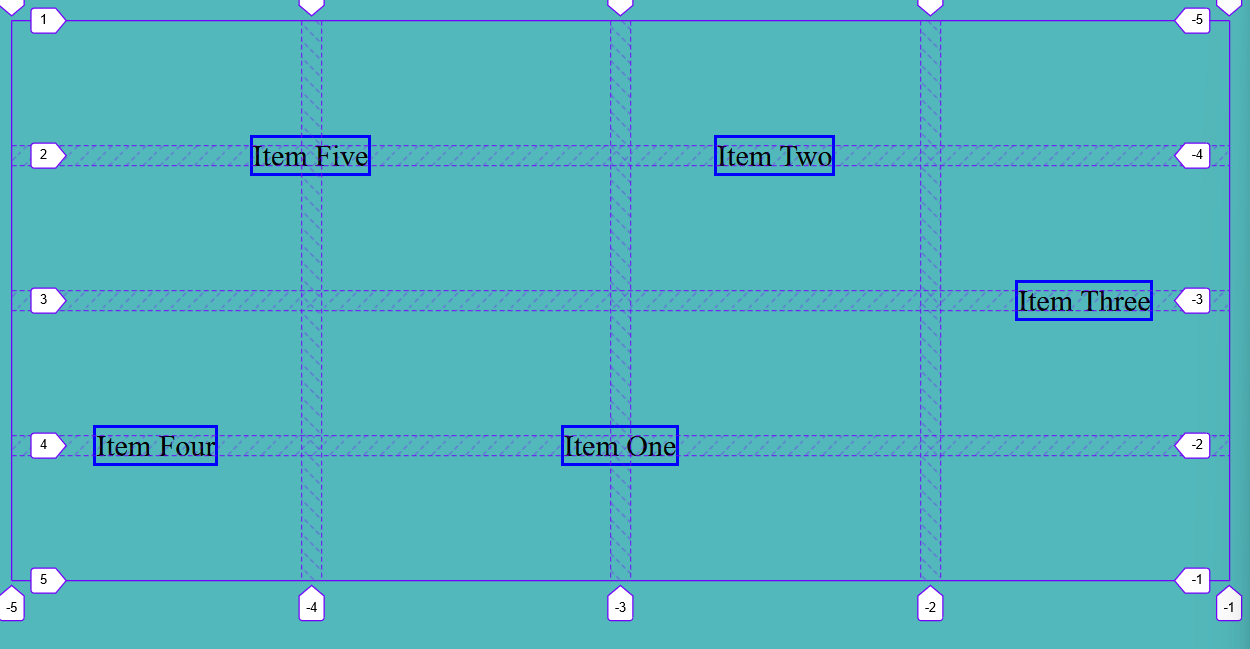


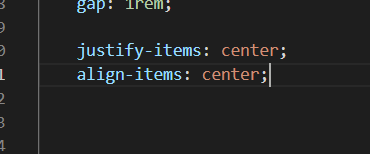
Here by default



And we can change as per our need

like this



****

But as we can see doing that changes our layout badly, so use as per situations

**Think of it**

**w.r.t the grid-container all inside it are the items**

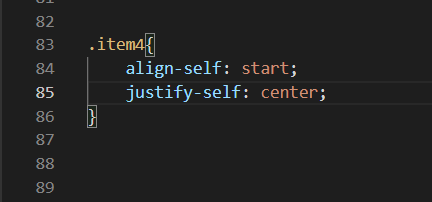
**justify-self and align-self**

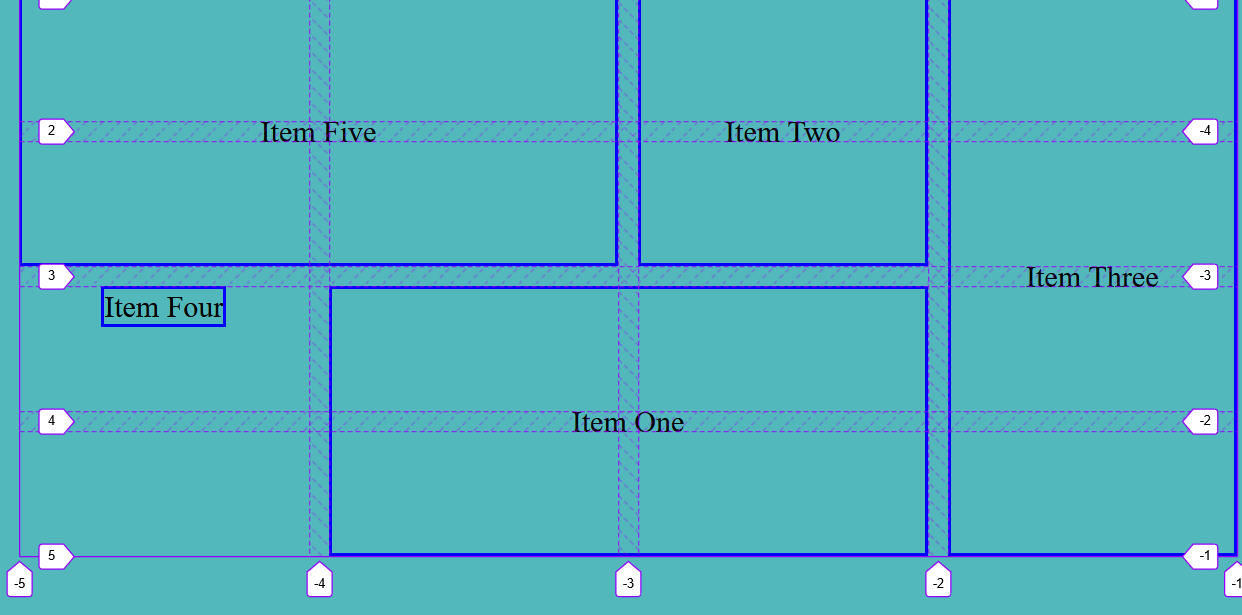
Similar to above two but these applies to each grid item

Meaning we can have different values of these properties for different items

here also the justify-self applies along main-axis / along x-axis / along right-left

and align-self applies along the cross axis / along the y-axis / along top and bottom



****

**Designing a 12 column grid, a 12 column one is popular**

Since 12 is divisible by most numbers so it’s best to design in 12 column grid system

But we can always design in any grid system

Remember

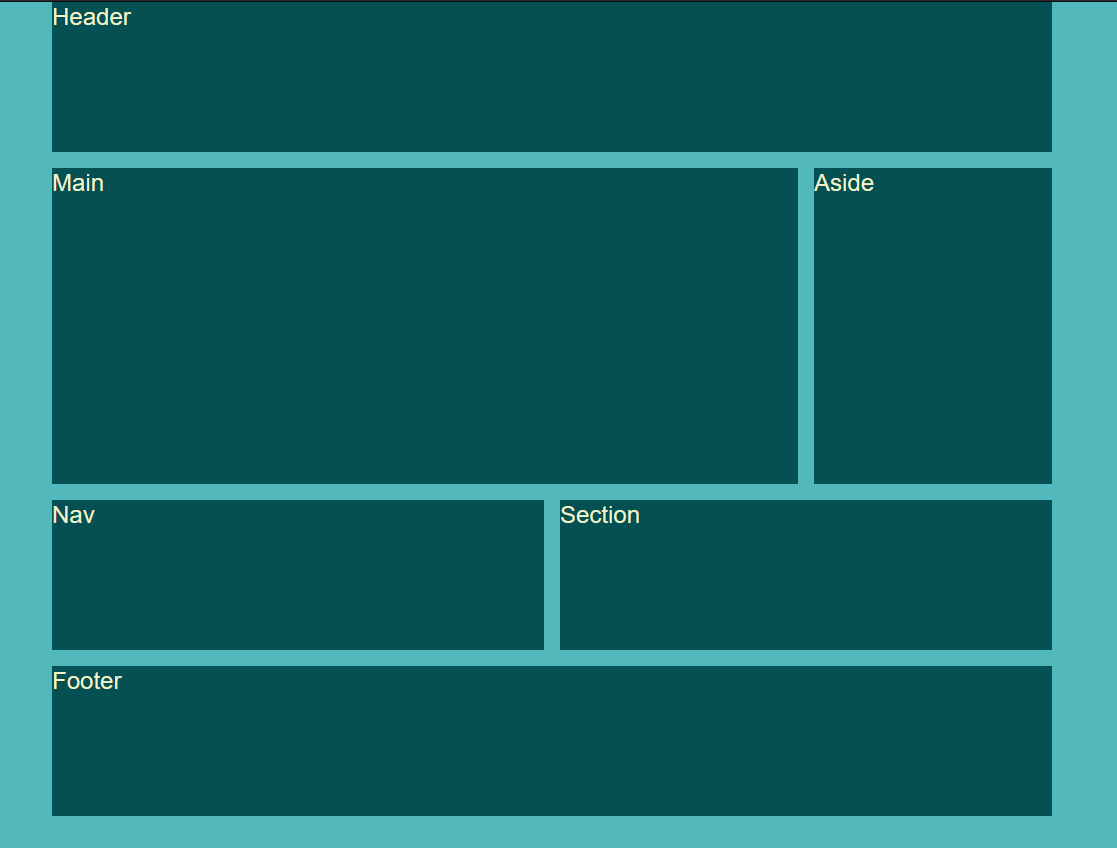
Grid maa let say we do

.grid-container aside {

grid-column:9 / 12;

grid-row:2 /6; // We mostly let row to be generated automaticly so on doing this let say last row line ends at 4, then due to above property new rows will be automaticly created

}



**Let say we want to took out writing content to the center**

doing

align-items and justify-items in the grid-container won’t do any good, because w.r.t the grid-container the items are the <header> <main> <nav> <footer> not

what is written inside them

Also doing align-self, justify-self doing in each tags also won’t do any good since they themselves will get arranged not content inside them

To adjust the content inside them we have to do

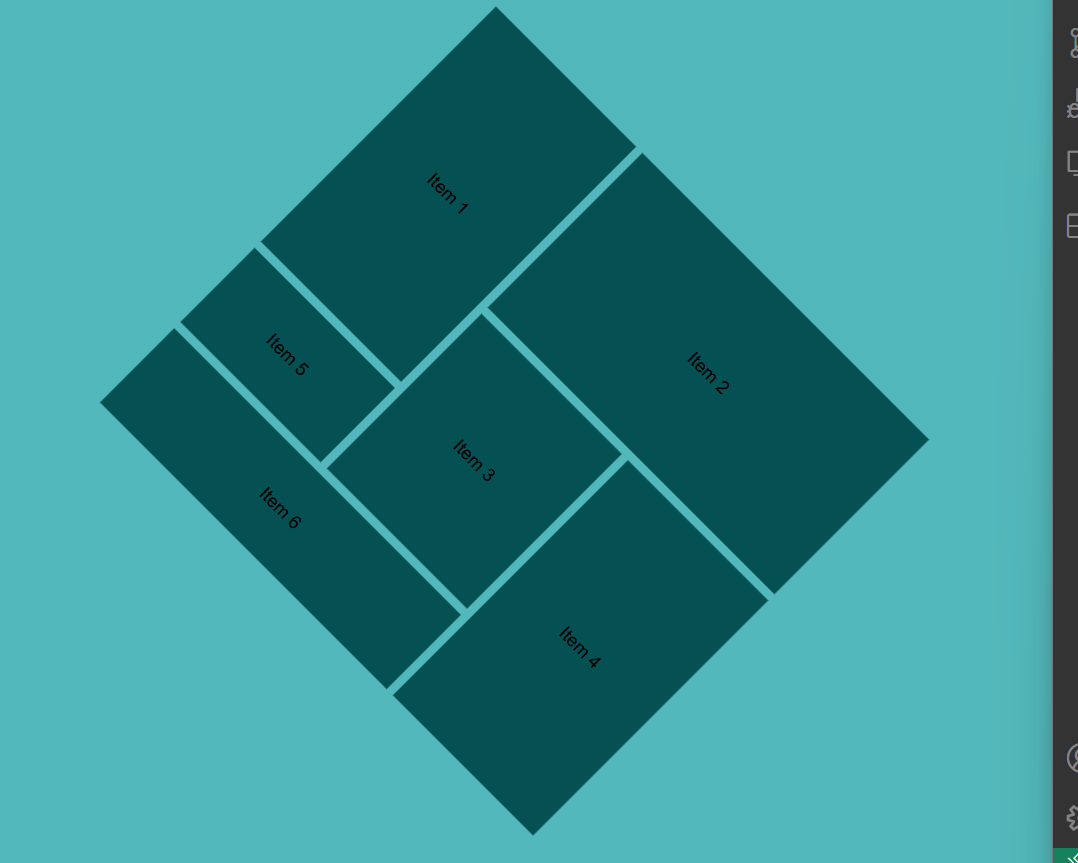
either

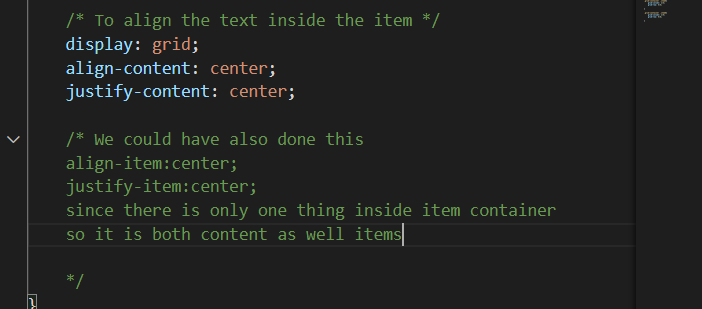
display:grid;

align-items:center;

justify-content:center;

or by flex-box method same as always





item bhitra eauta marta content xaa

so it is both

content

as well as

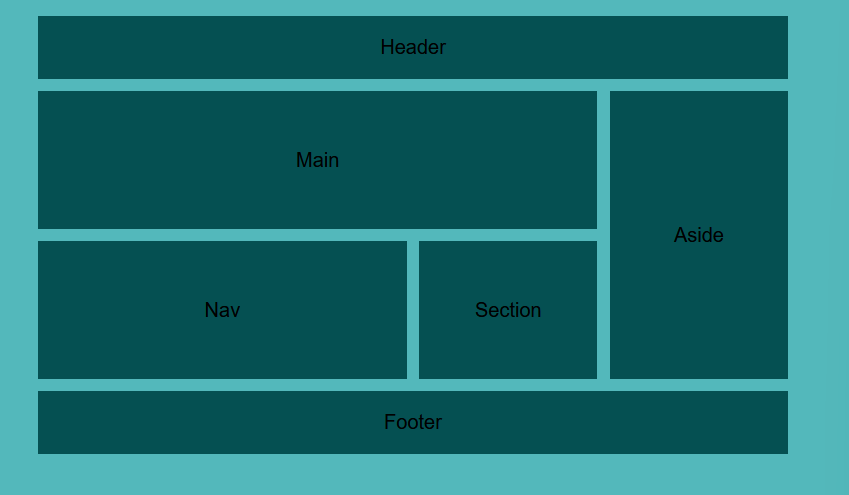
items

so both properties works

**content- vako properties lae whole content lai align garxaa**

**items-vako properties lae items lai align garxaa**

**grid area is also very handy, easy and powerful**

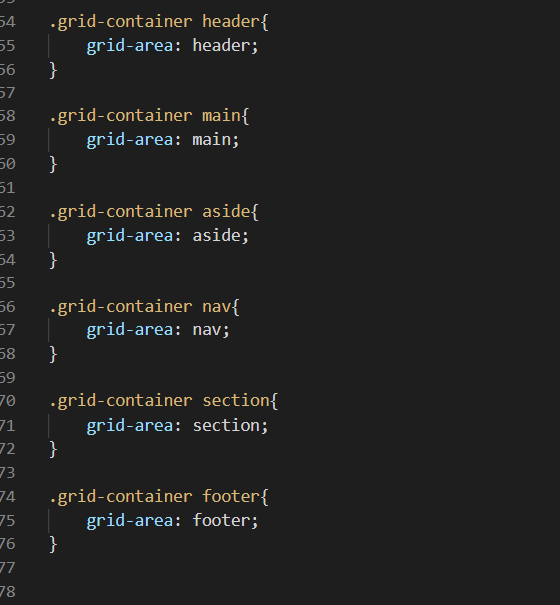


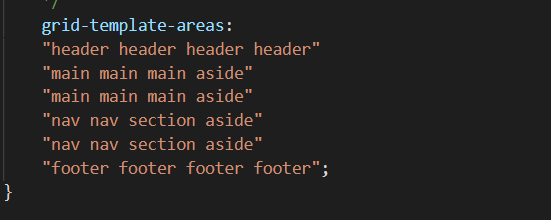
We can create layout like this very easily using the grid

We just need to first name the each grid item, as grid-area

than we use

grid-template-area: in the main grid-container





See seems like we are writing CSS just like layout

Here the css layout is written considering the column number

**Media Queries**

