# 2) Using the Bootstrap 4 Grid

Bootstrap grid is simply created by bunch of html elements like typically divs. You have one div with container class. In this with div we now create rows and columns to define our grid because that is how all grids typically work. Then we add another div with row class. We can add as many rows as we need. In rows classwe add columns and columns are places where you put your actual content. Columns can be as wide as you will learn. You essentially use them to create cells in your grids. You got rows with columns well you got cells. Columns in its simplest form is crested by adding col class.

Here is our code-

<div class="container">

<div class="row">

<div class="col">

Row 1 Col1

</div>

<div class="col">

Row1 Col2

</div>

</div>

<div class="row">

<div class="col">

Row 2 Col1

</div>

</div>

</div>



Now I want to style it, for this I will add my own classes. We add my-cntainer,my-row, mycol classes.then we define these styles after bootstrap import in index.html

New cde-

<style>

body {

margin: 30px;

}

.my-container {

border: 1px solid green;

}

.my-row {

border: 3px solid red;

}

.my-col {

border: 3px dotted blue;

}

</style>

</head>

<body>

<div class="container my-container">

<div class="row my-row">

<div class="col my-col">

Row 1 Col1

</div>

<div class="col my-col">

Row1 Col2

</div>

</div>

<div class="row my-row">

<div class="col my-col">

Row 2 Col1

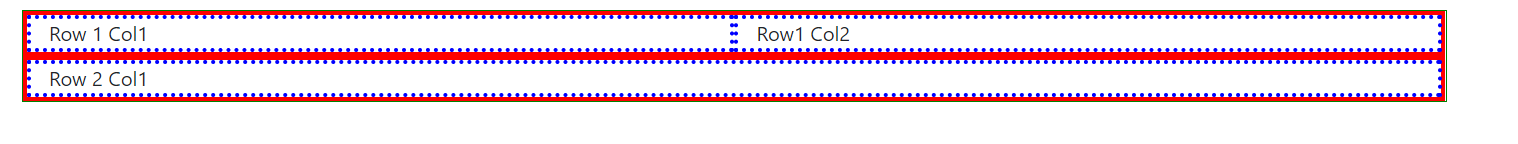
</div>

</div>

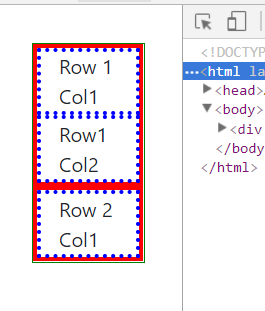
</div>

</body>

</html>



Here ech columns takes equal amount of space, if we resize the page again each column will take 50% of total width. If I shrink the scrren size, again 2 columns take equal amount of space. However if we really push it, then all columns come under one columns. We have to go below a threshold.



I have to go really-really small for that. So this is default behaviour, if I just use col class. Generally we want columns of different widths. We can do this by tweaking col class and by adding additionally info. Bootstrap has different col classes. Now we use col-8 and col-4 classes in first row. These numbers here are shares of avalaible space. They always have to, and that’s important, add upto 12.to be more precise, they can add upto 12-that’s the maximum. but you can also use less than12 as a sum. We get 12 column grid by default. With col-4 we told bootstrap that this column should have width of 4 columns(if we deivide grid into 12 equal column, then col-4 means give it width of 4 columns). We can also use col4 and col6. Then we can leave 2 cols empty at the end of the row. Sometimes we need to be more precise than that, because we want to have different layouts for small and big devices. lets say for small devices, i.e your mobile phones we want to stack these columns. We want them to sit beneath each other but for larger and medium devices we want them to be beside each other. we can get this by tweaking classes even more, besides giving the share of overall width a column should get, we can add size name so as to say, to define to which kind of screen size , it should apply. We can use col-md-4, this means for medium and large screen it should get 4 column’s width.there are 5 kind of classes avalible. We can have medium size screen with md, for large we have lg, xl for extra large. then on other side of medium , you got sm for small(tablet size) and xs for extra small(mobile phones).

<div class="container my-container">

<div class="row my-row">

<div class="col-md-4 my-col">

Row 1 Col1

</div>

<div class="col-md-8 my-col">

Row1 Col2

</div>

</div>

We used col-md-4 instead of clo-4. Now these will be beside each other only in bigger screen, if we decrease width then they come below each other. lets say instead of having stack we want to have these columns next to each other then we use a additional class, like this-

<div class="row my-row">

<div class="col-md-4 col-sm-6 my-col">

Row 1 Col1

</div>

<div class="col-md-8 col-sm-6 my-col">

Row1 Col2

</div>

</div>

Now for smaller screen these share eqal width. For larger there will be no change. However if again we go to very smaller screen they again stack on each other. but now we have to go to further smaller screens to get them stacked on each other. this is how we can use numbers and width classes togather.

This is basics we can do more. Lets say we have column widths that do not add upo 12. So we will get empty space, in row 2.

<div class="row my-row">

<div class="col-4 my-col">

Row 2 Col1

</div>

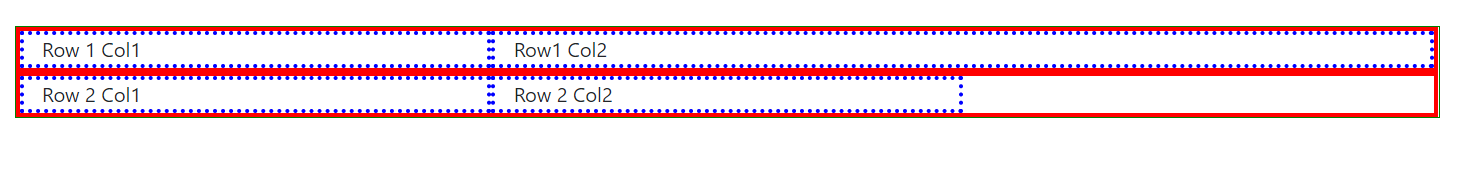
<div class="col-4 my-col">

Row 2 Col2

</div>

</div>

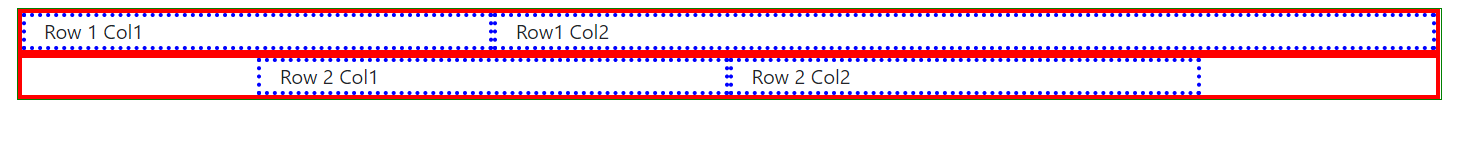
Now they will take equal space but they wnt take up full width because they do not add upto 12.



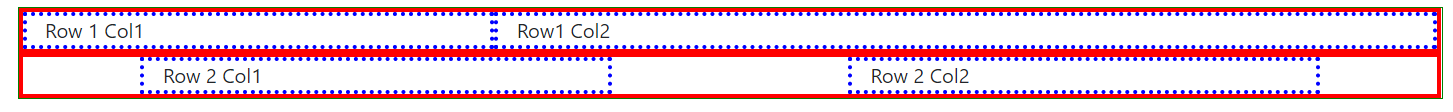
Lets say we want to center 2 columns in row 2,horizontally. For this you can add additional class. This class needs to added to div with class row. Class here is inspired by flex box setting that are happening behind the scenes. So this class name will look familiar if you know felxboxes. So we add this class-

<div class="row justify-content-center my-row">

Justify content is flexbox rule for setting where content should be positioned. now content is centered-

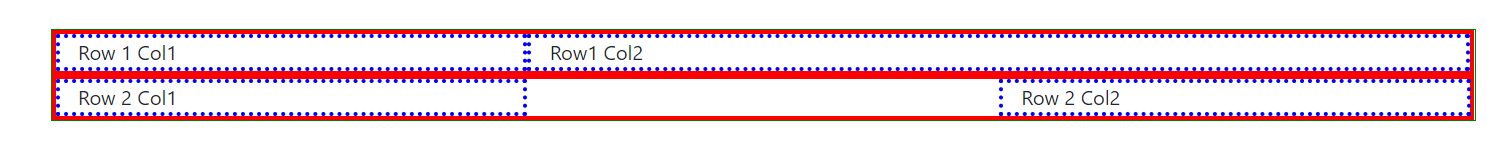


Instead of center you can also use **start**(this is default, you dnt need to specify it)and **end**. Start move them to left and end moves them to right. We also got **around**. It will basically add some space between , (space around them to be precise )columns, so it will put space to left,right and between them. With around-



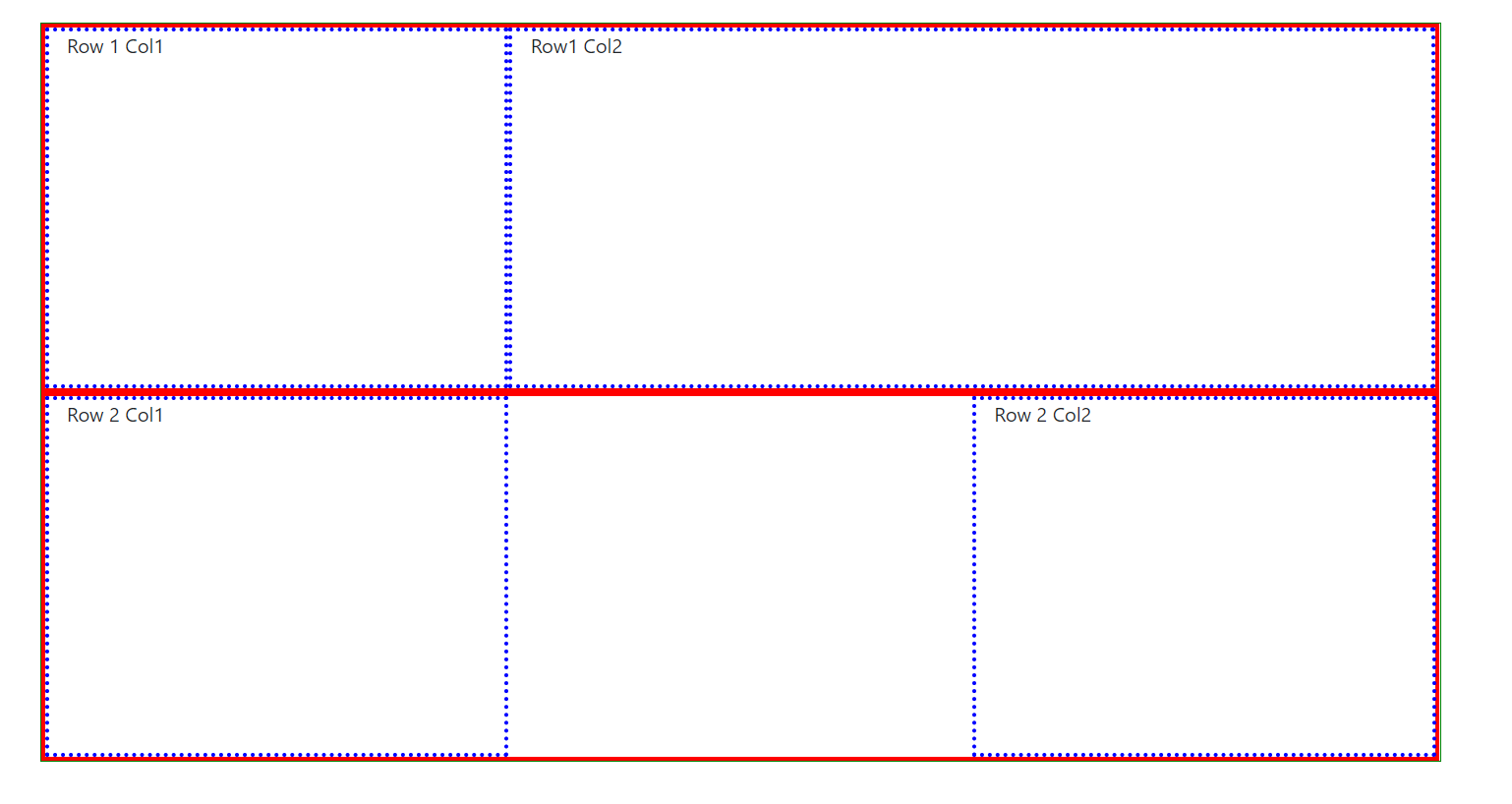
Now they are kind of centred but there is space between them. So empty space is divided equally to left, right and then to left and right of each column, making it double in between them.

Some times we just want to push columns to their near end, for that use, **between**.



This puts all the available space between 2 columns. So all these classes make alignment on horizontal axis very simple. What about vertical alignment. Vertical alignment has been very difficult in css. But with flexboxes its easy, so its also easy in bootstrap.

First lets give our row a height. To my row class we give height of 300px. Now columns take all height by default-



Lets say we do not want colums to take full height. We want them to take height that you set for the columns or content requires. Therefore we can also define how they should be positioned vertically. So we got to second row, in addition to **justify-content**-something we can also add **align-items**-something, again coming from flexbox property used behind the scenes. We can use start for example-

<div class="row justify-content-between align-items-start my-row">

<div class="col-4 my-col">

Row 2 Col1

</div>

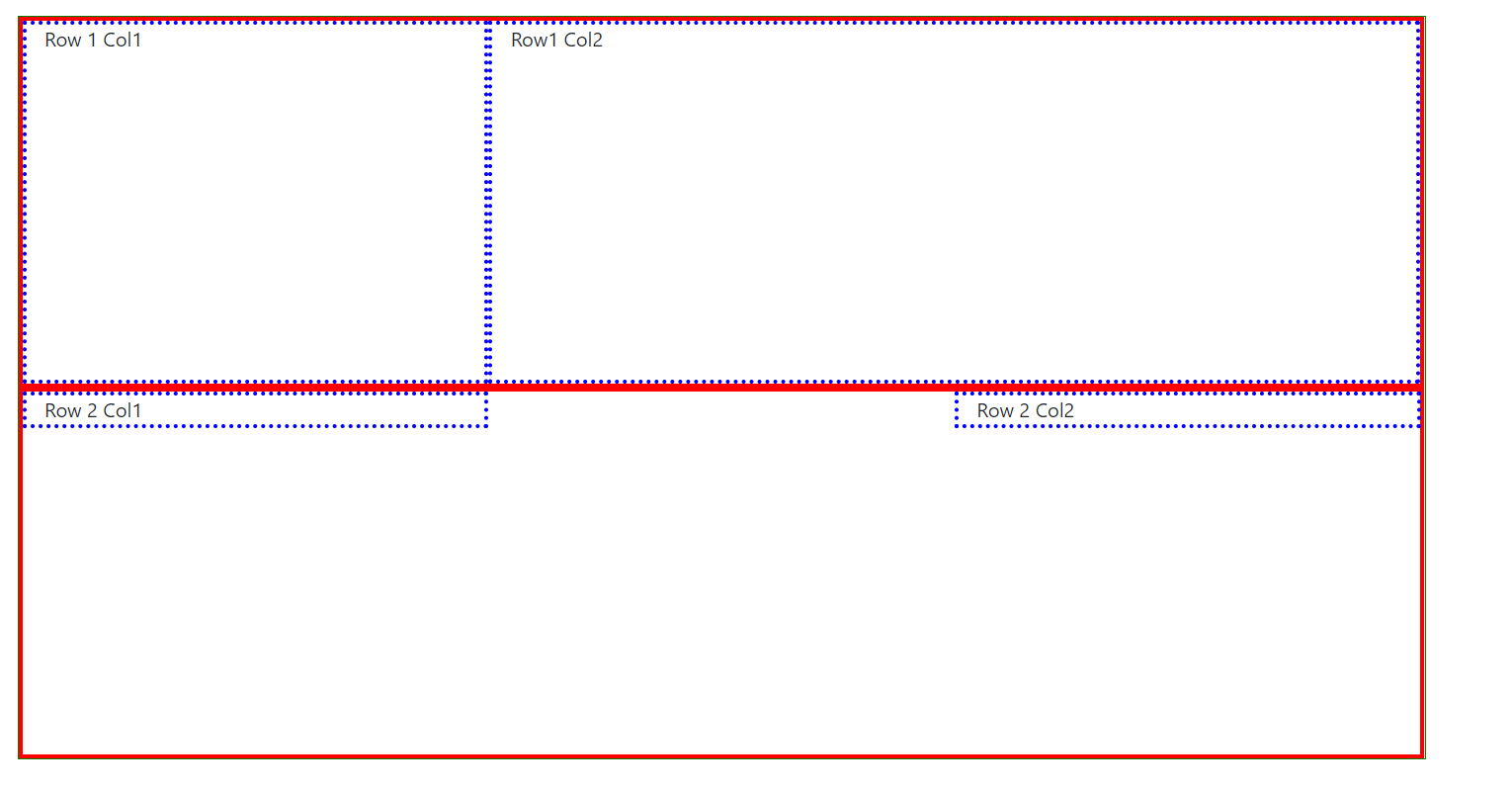
<div class="col-4 my-col">

Row 2 Col2

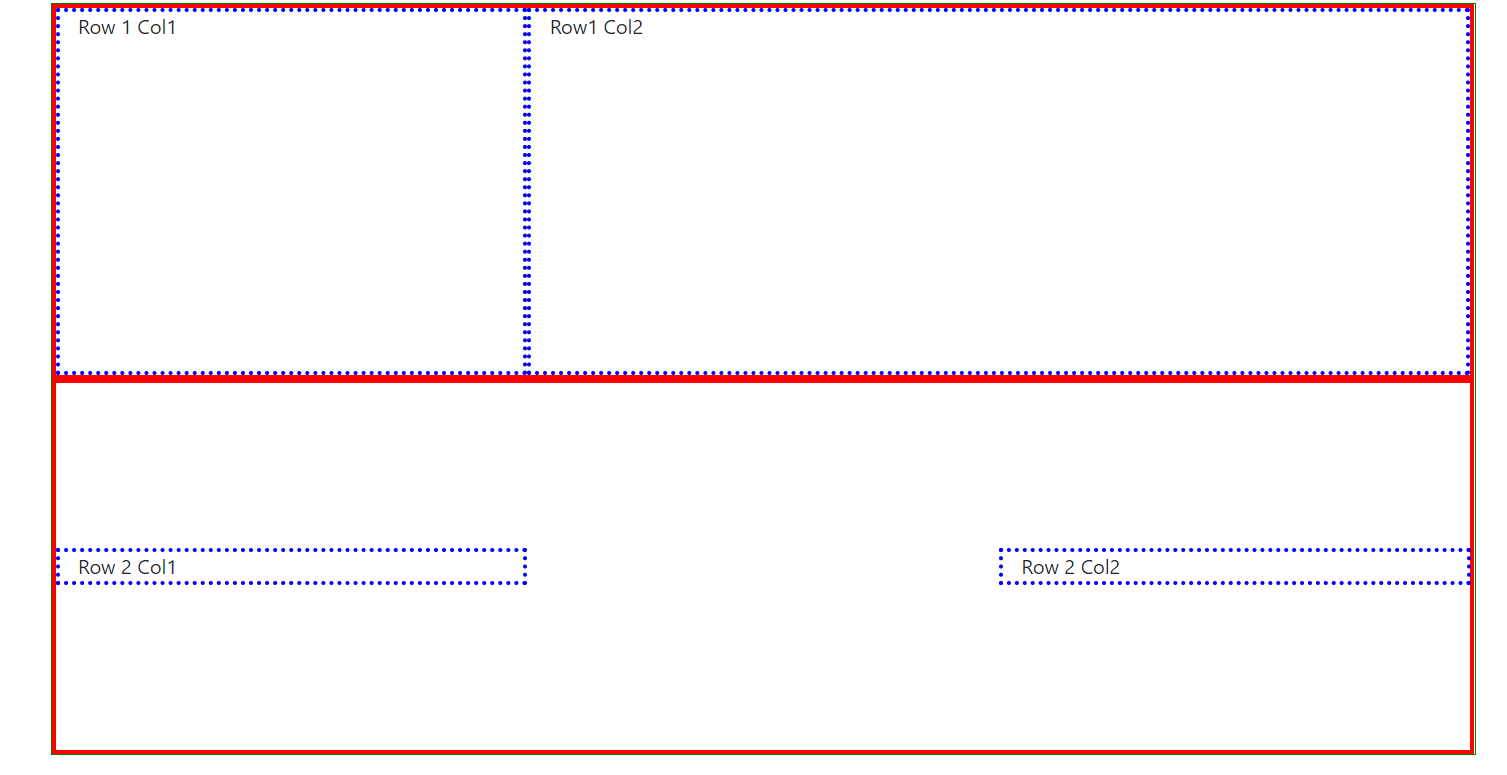
</div>

</div>

Now columns will be only as high as required by their content and they are pushed to start , (vertically )of their row. This looks like this-



We also got **center** and **end**. They basically change the position of col, but height remains same. You can guess how they work. This is what center does-



We also have **stretch.** This is default, I mean this is applied even if we dnt use **align-items** class. But we have been using these classes on row, which means all columns will behave like that. Lets sometimes we want to change the behaviour for one column and not for all.there fore we need to apply them on div with column class. On columns we use this-

**align-self-something. Code-**

<div class="row justify-content-between my-row">

<div class="col-4 my-col">

Row 2 Col1

</div>

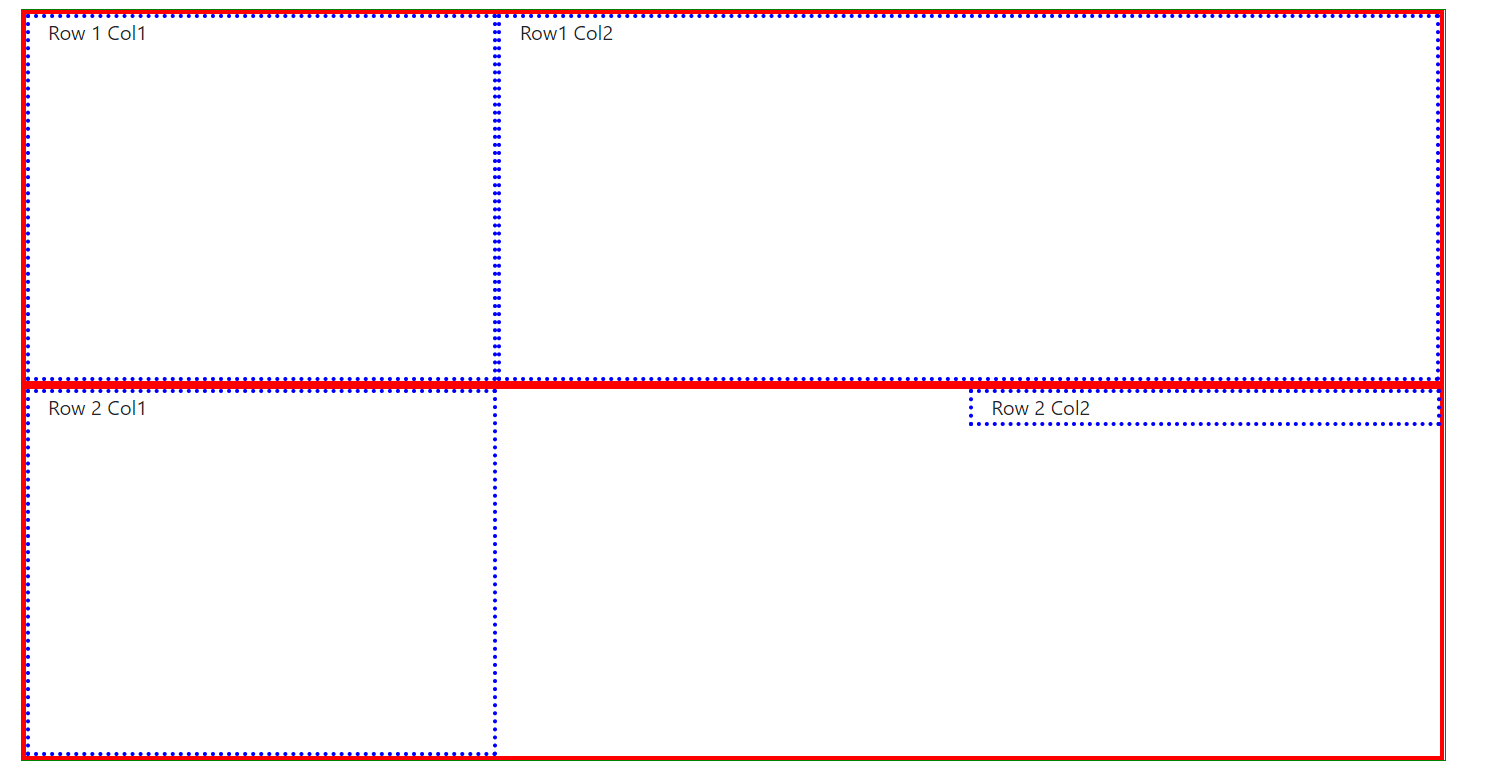
<div class="col-4 align-self-start my-col">

Row 2 Col2

</div>

</div>

**Browser-**



now second column is pushed up while first says where it was.

Go through official docs, given in description to learn more about grids-

<http://getbootstrap.com/docs/4.0/layout/grid/>

Lets see one more thing, how to switch order of columns. You should be careful with this, it’s possible as you will see in second, but it’s very confusing. Sometimes it’s very useful,like we want different order for our desktop and different order in mobile. We can do this adding class to column div. here we attached **order-12** class to our first column. This goes all the way upto 12 because we got 12 column grid .then to second column we give **order-2**(or any other class smaller than that)

<div class="row justify-content-between my-row">

<div class="col-4 my-col order-12">

Row 2 Col1

</div>

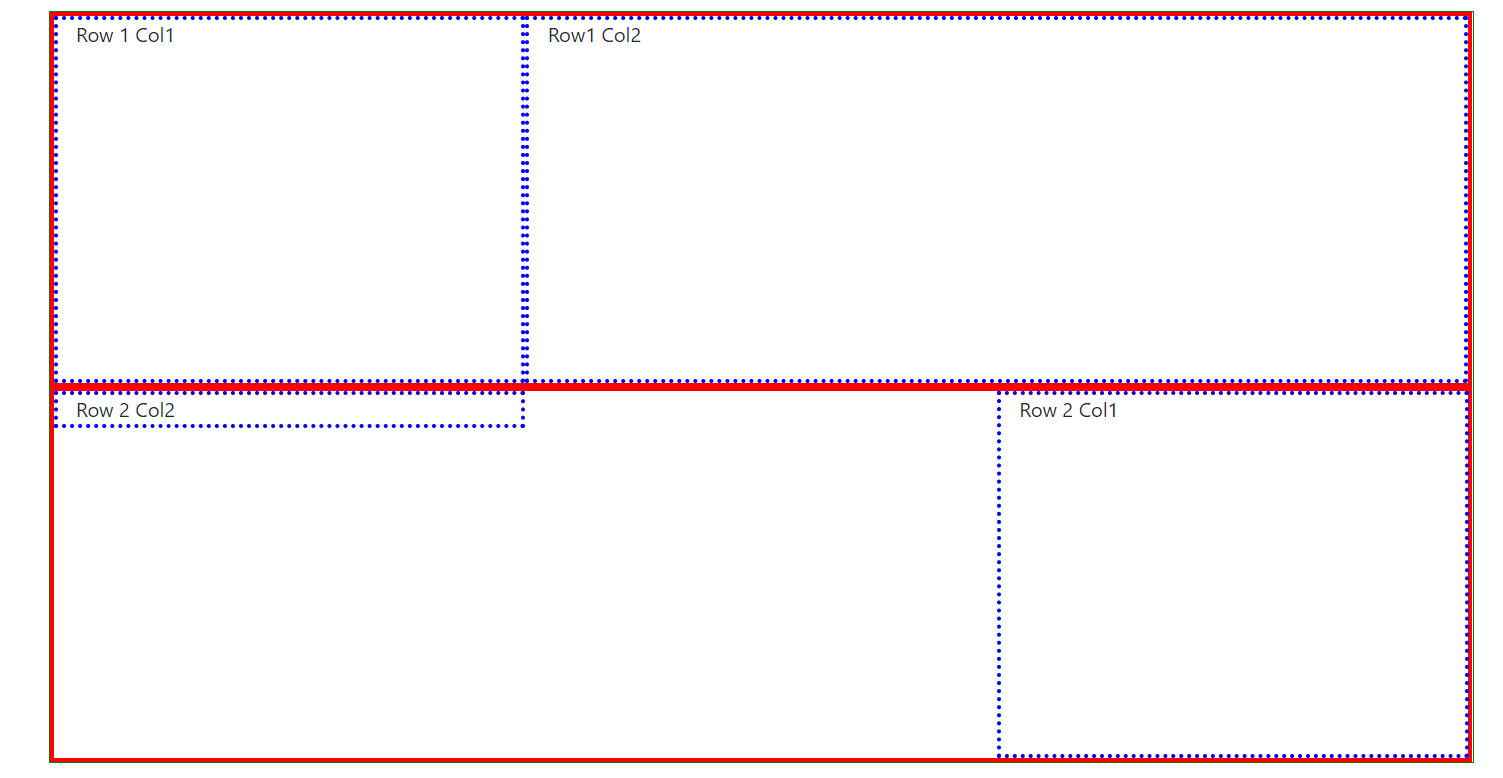
<div class="col-4 align-self-start order-2 my-col">

Row 2 Col2

</div>

</div>

In browser we can see that column 2 comes before than column-



It is because order of numbers matters here. Lower number wins and moved to left, bigger number is moved to right. If we have more than 2 columns then also order number will be taken into account. Now this can be used in conjunction with device size breakpoints. Lets say we want this order to be followed for bigger screens only. Then use- **order-md-12** and **order-md-2** class. For other devices default order will be used.

<div class="row justify-content-between my-row">

<div class="col-4 my-col order-md-12">

Row 2 Col1

</div>

<div class="col-4 align-self-start order-md-2 my-col">

Row 2 Col2

</div>

</div>

Now whenever we shrink the size the order of columns changes. This is great for creating responsive layouts where you maybe want to switch some elements to top for your mobile devices. To do this we will write code in mobile first way i.e write the code as we want to have in mobile, then we will switch the order by using order class for medium devices.

Last thing that I want to show you is offsetting, this means pushing or moving your column to left by certain column count which you can kind of achieve with **justify- content-between** or around part. but there avalaible space is split up evenly. Now if you want to say, I got 2 columns with width 4 and I want to move them to left by, lets say 2 columns by default and then only treat the remaining space thereafter as remaining space. Then we can **offset-2** class to our columns. This simply tells bootstrap that move this to 2 columns to right.

<div class="row justify-content-between my-row">

<div class="col-4 offset-2 my-col order-md-12">

Row 2 Col1

</div>

<div class="col-4 align-self-start order-md-2 my-col">

Row 2 Col2

</div>

</div>

Here you wnt see any difference because we switched our order and it will overwrite this but if we decrease the screensize we will see that first column has moved to right by 2 blocks. Now we can also make this offset class responsive by adding breakpoint names.

**offset-md-2**  means push it by 2 blocks right only on medium sized devices. Like this-

<div class="row justify-content-between my-row">

<div class="col-4 offset-md-2 my-col order-md-12">

Row 2 Col1

</div>

<div class="col-4 align-self-start order-md-2 my-col">

Row 2 Col2

</div>

</div>

However we do not see any change for medium size device.e offset is not applied, that is because order thing here actually overwrites this. To be very precise order does’nt override offsetting , keep in mind that we switch the order and we offset this first column(we apply offset class on first column), now this is first column only in mobile device so we also need to add this offset code to second column, infact we only need to add it there, because second column in code will be our first column in medium size device due to this order thing.