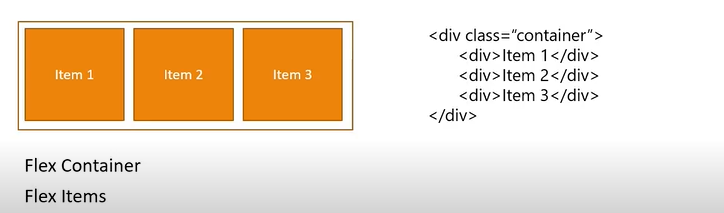
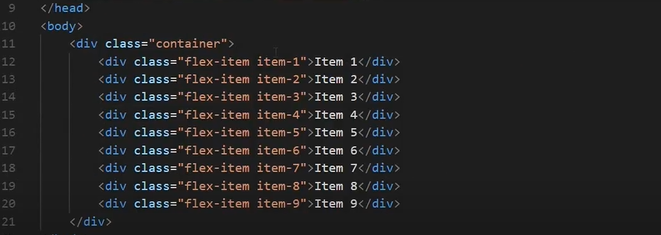
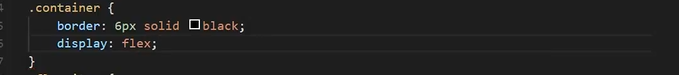
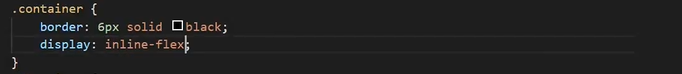
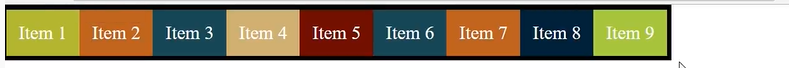
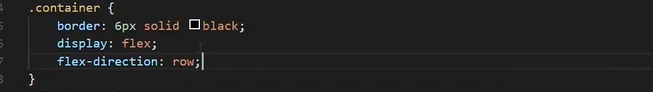
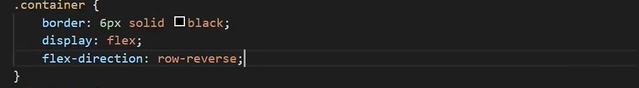
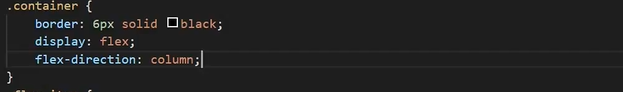
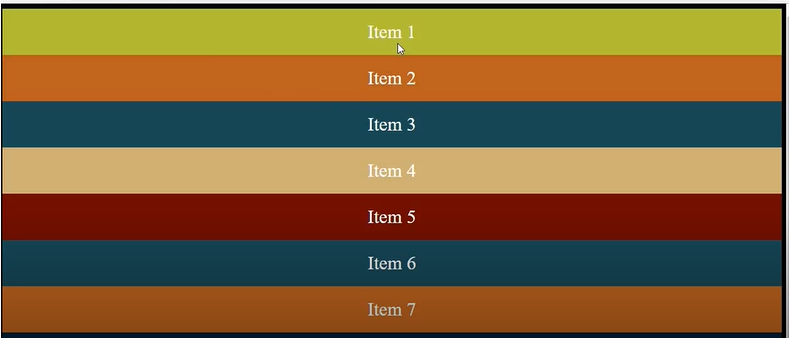
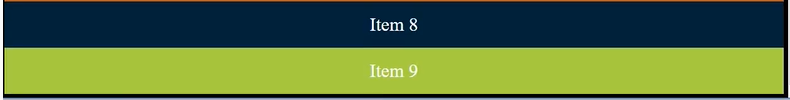
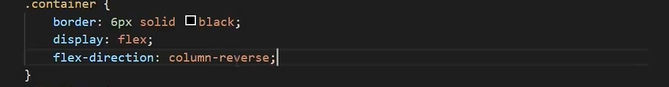
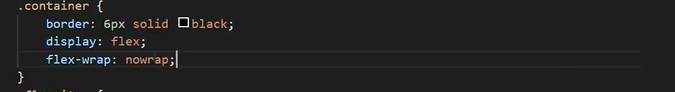
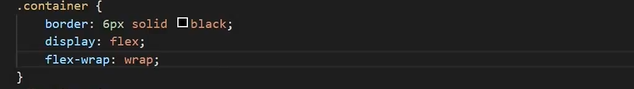
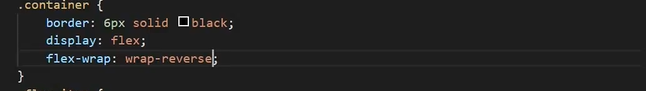
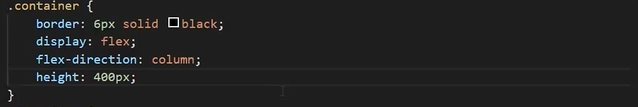
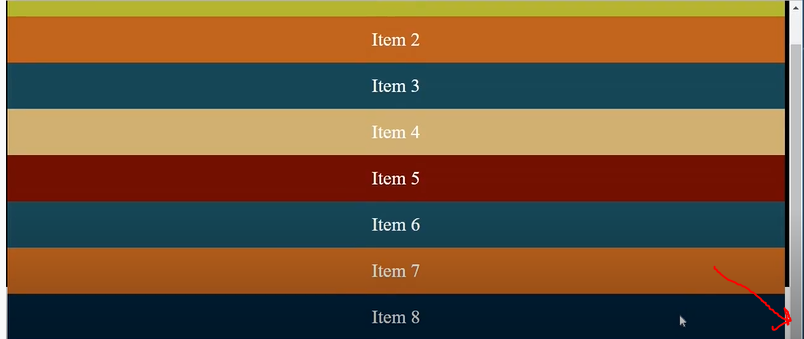
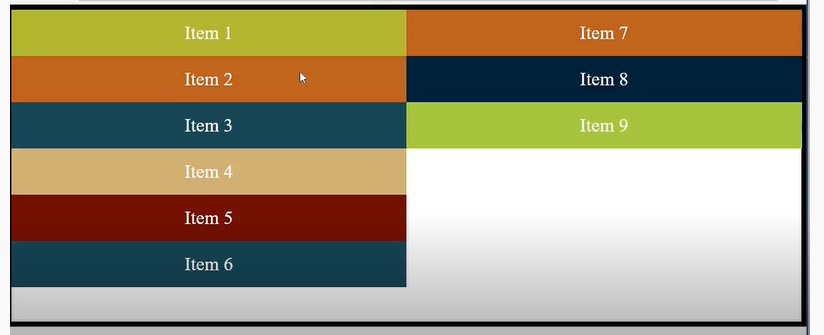
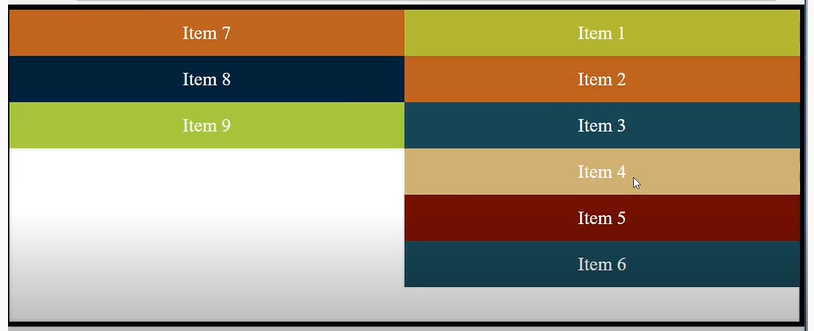
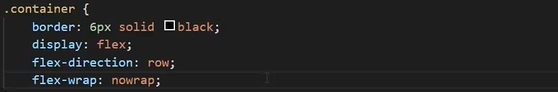
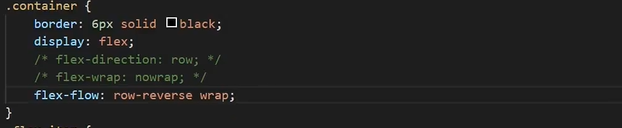
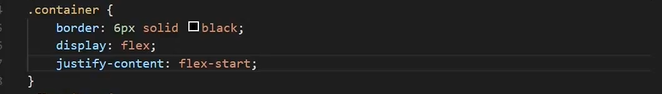
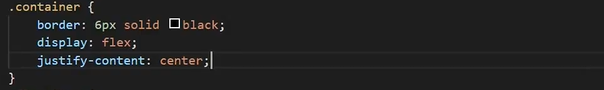
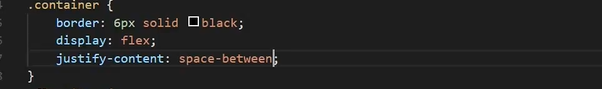
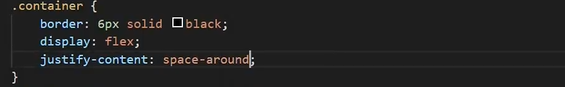
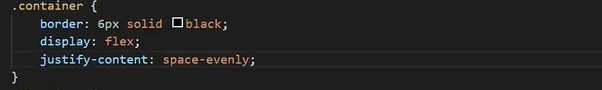
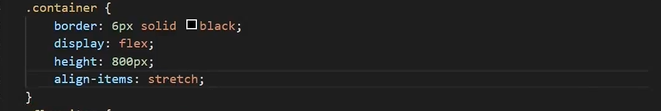
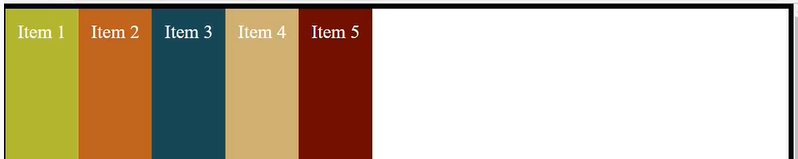
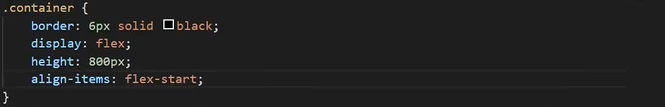
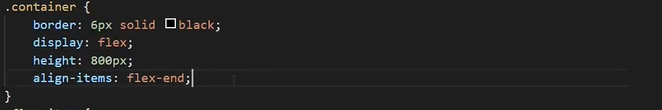
**FLEXBOX**what is Flexbox ?  
CSS Flexible Box Module is a one-dimensional layout module that makes it easy to design  
  
  
  
so before Flexbox there are four Layout modes   
  
  
  
But these Layout did not provide enough flexibility on the Flex Layout provides a lot of flexibility.  
with Flexbox we can arrange items left to right, right to left, top to bottom or bottom to top and  
at the same time control the spacing alignment and order of items in a container beside this bootstrap 4 is built on top of the Flex Layout.  
  
Now we will understand the Flexbox and how it will works.  
  
In Flexbox we have two entities a parent container term as the flex container and the immediate children elements which we term as flex items.  
  
  
  
If we look in the above figure the parent div is flex container and all the children divs are flex items .  
  
we also have two axes while working with the Flexbox.  
1) Main Axis.  
2) Cross Axis.  
  
By default the **main** **axis** runs from left to right and the **cross** **axis** is perpendicular to the **main** axis and runs top to bottom. The starting point of the main axis is termed as the **Main** **start** and the end point is termed as **Main** **end**. The length from Main start to Main end is called the **Main Size**.  
  
so we can say that the Flex items flow from main start till main end and take up the main size as length similarly with cross axis we have cross start, cross end and cross size. We can also change the  
the direction of main and cross axes.  
  
  
  
  
**Flex Container Properties**1) display   
 This defines the flex container and is mandatory to work with Flexbox.  
2) flex-direction  
 It defines the direction in which the flex items are placed in the container.  
3) flex-wrap  
 It is used to control the wrapping of items with in a container.  
4) flex-flow  
 It is a short hand for the combination of flex direction and flex wrap.  
5) justify-content  
 It defines the alignment of the items along the main axis.  
  
  
6) align-items  
 It defines the how flex items are layout along the cross axis.  
7) align-content  
 It is similar to justify-content with the difference being this will align along the cross axis instead of   
 the main axis also the align-content works only when there are multiple rows of flex items in the   
 container.

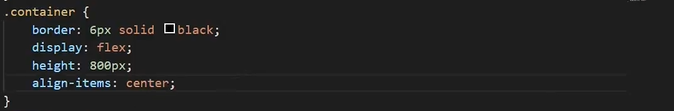
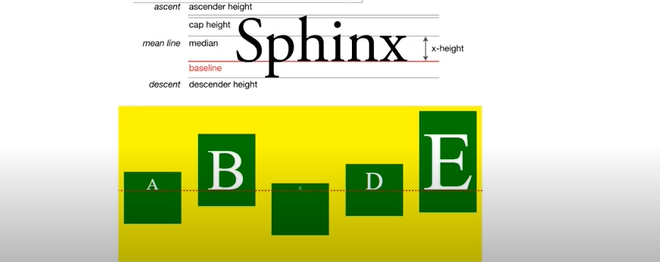
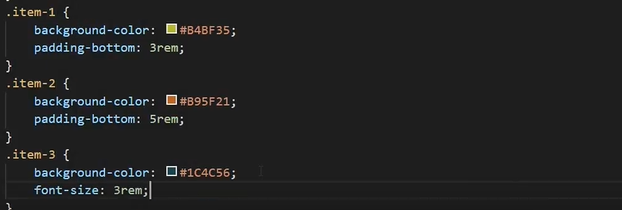
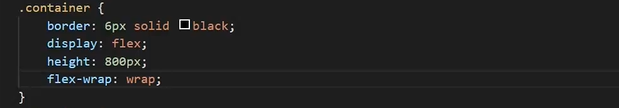
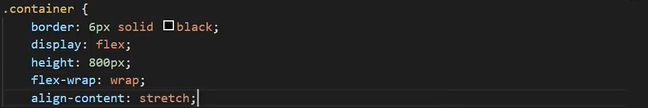
so, we will discuss each property separately.  
 1) **flex display** we will create two files index.html and style.css so consider a part of the index.html file below  
   
   
  
In the body we have a parent div tag that has a class container now this parent container 9 divs as children each div tag has a class flex-item and also a class specific to the item number item-1, item-2, item-3 and so on .  
  
Now coming to the stycle.css and consider a part of style.css file below  
  
  
  
First we set a margin 0 on the body next on the container next on the container class we set a really thick border because it is important that we are able to identify the boundaries of the container so the border is 6px solid black   
  
The flex-item class has font color, font-size, padding and text-alignment properties each of the individual flex item classes have a background color to visually separate the items in the browser.  
  
so when we open the index.html file in the browser we get the below.   
  
  
  
we can see that we have 9 items enclosed within the container that has a solid black border so now let us focus on very first property i.e display  
  
To create a flex container we make use of the display property and set it to value of flex.  
  
so in the css file in the container class add below.  
  
  
  
so now when we refresh the browser we get the below output.  
  

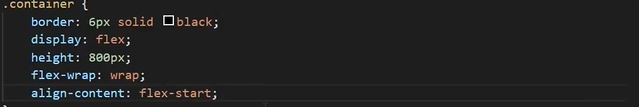

The items are lay out from left to right instead of traditional way where div tag stack up from top to bottom also you can make out from the border that the container has the 100 % width so it does behave similar to a block level element and if you don’t want a block level flex container then you set the display property to inline flex to create inline flex container. So change the property from flex to inline flex like below in the css file.  
  
  
  
and we get the output as below. Look at the previous border and current border below.  
  
  
  
see that the border now wraps around the flex items. The container only takes up the enough width to accommodate its children make sure display property is always set or none of the other properties are going to work on the flex container.  
  
so display property create either a block level or inline level flex container and the possible values are flex and inline-flex.  
  
2) **flex direction**  The flex direction establishes the main access which in turn besides how the flex item are placed   
 inside the flex container so by default the main axis flows from left to right which is the reason we   
 see the items placed from left to right in our example so by changing the value of flex direction   
 property we can change how the items are placed.  
  
so make below changes in the css file for the class container i.e add the property flex- direction like below.  
  
  
  
There for we can put four values for flex direction first value is row which is by default value. So it does not make any change in our current output as the flex direction of value row sets the main axis from left to right.  
  
Next value is **row-reverse** it sets the main axis direction from right to left which results in flex items being placed from right to the left.   
  
  
  
we will get below output for the above.  
  
  
  
we can see the items are placed in reverse order.  
  
Third value allowed is column in which the main axis flows from top to bottom.  
  
  
  
so the items are stacked on top of each other .  
  
  
  
  
so item 9 is at the top and item 9 at the bottom so the items are stacked and flow from top to bottom.  
  
Now if we have the column-reverse value of flex direction like below in the code.  
  
  
  
the we have item 9 at the top and item 1 is at the bottom so item flows from bottom to top.  
  
  
  
3) **Flex wrap**  
 By default all the flex items in the container will try to fit into a single line if there is not enough   
 space then the items simply overflow. so suppose we have below in the browser width.  
  
  
  
But if we reduce the browser width then items may shrink and no longer in the view like below.  
  
  
  
so with the help of flex-wrap property we can change this behaviour. So flex wrap property can accept three values.  
  
1) nowrap  
 This is the default value  
  
  
  
so by applying this property there will be no change in the output of the browser and it is same as the previous one when we reduce the browser width.  
  
2) wrap  
   
  
so when we apply this property we see changes in the browser like below.  
  
  
  
so you can see the items wrap in the container when we reduce the browser width if there is no enough space for just one item here item 9 example so only item 9 moves to the next row.  
  
3) wrap-reverse  
 Instead of items flowing into the row below it climbs into the row above.  
   
  
 so we will get below output when we reduce the browser width.  
  
  
  
wrap-reverse just pushes the last item above instead of below .  
  
so similar horizontal wrapping vertical wrapping is also possible so for that we need to change to flex-direction to column and add height to the flex container like below in the code.  
  
  
  
we get the below output if we observe very carefully the border ends at the item 7 and rest of the items are overflowing the container.  
  
  
  
so if we add the flex-wrap and add a value of wrap like below   
  
  
we get the below output in the browser we can see the items are wrapped with in the container.  
item 1 to 6 fit in the one column then rest items move to the next column i.e column created to the right.  
  
  
  
  
similarly we change the value of flex-wrap from wrap to wrap-reverse the extra items are placed to the left hand side like below in the code.  
  
  
  
4) **Flex flow** It is short hand for flex direction and flex wrap. By default flex-flow property is set to row and   
 nowrap which is the default of the individual property like below .  
  
  
  
Now we can comment out these two properties and add the single flex-flow property where first value is flex direction and second value is flex wrap like below.  
  
  
  
Initially it is like in the browser  
  
  
  
 but after applying the above properties if we look in to the browser we get the below output.  
  
  
  
so we see that row reverse has applied so item 1 is to the right and item 9 is to the left and if we start to reduce the browser width we get the below in the browser  
  
  
  
Items start to wrap item 9 moves to the next line then item 8 then item 7 and so on like this.  
  
so now let us try column- reverse to the container like below.  
  
  
  
we get the below in the browser  
  
  
  
item 1 at the bottom column reverse and then wrap reverse the items are wrapped to the previous column so basically flex-flow  
  
  
  
5) **justify content**  
 It defines the alignment along the main axis By default it has value of flex-start which places the   
 flex items at the beginning of the main axis which is also known as main start.  
 so we have reduced the no items from 9 to 5.  
  
  
  
Initially we have this in the browser.  
  
  
  
so after above property there will be no change in the output.  
  
if we set value to flex-end and it will cause flex items to be placed at the end of the main axis so get below output in the browser items are pushed to the right which is the end of the main axis  
  
  
  
It is also possible to align the content at the centre of the main axis.  
  
  
  
we get the below output in the browser so content is placed at the centre of the main axis.  
  
  
  
However we also properties that how to distributed the extra spaces if we give the value space-between like below.  
  
  
  
we will get below output in the browser. we see that extra spaces that was left out evenly split and added in between the flex items.  
  
  
  
sometimes we want the space before the first item and after the last item so for such a scenario the value will be space-around like below.  
  
  
so we will get the below output. So now there is a space at the beginning and at the end and space is equal to half of the space between the flex items   
  
  
  
However if we want the same space even at the start and at the end so value be space-evenly like below.  
  


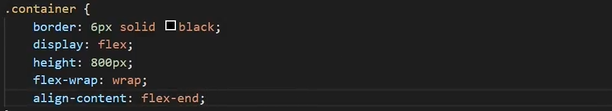
We will get below output in which extra space is equally distributed in the container

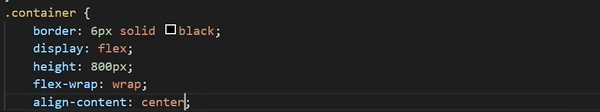
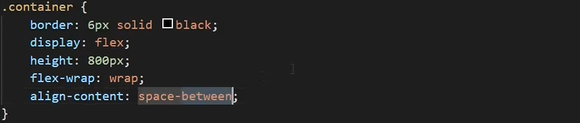
  
  
 so, if set flex-direction to column so justify-content deal with the vertical alignment.  
  
6) **Align items**  
 It defines the default behaviour for how flex items are lay out along the cross axis of the container   
 means it works like the justify content but in the perpendicular direction.  
  
 so by default the value of align items will be stretch it means the items are stretched to the entire   
 length of the cross axis.  
  
   
  
so look output in the browser   
  
  
  
so now putting the value flex-start  
  
  
  
so we will get below output from the previous one all the items are pushed to the cross start which is the starting point of the cross axis as we know cross axis flow from top to the bottom.  
  

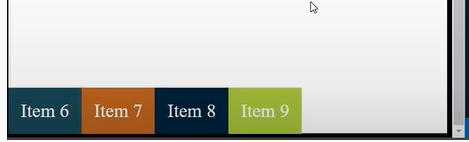
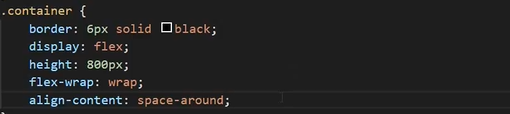

So when value will be flex-end like below so this value pushes the items to the end of the cross axis all the way to the bottom   
  
  
  
  
  


so, when value is set to the centre.  
  
  
  
which centres the context along the cross axis.  
  
so last value is baseline  
  
  
  
so when there is a word for example sphinx so the baseline is the line upon which the most letter sit so the baseline is the highlighted in the red line in the above image.so if you see in the images the we have 5 letters with different contents and different heights so just looks how all the items are sit on the baseline.  
  
The baseline value for flex items will align the flex items along their contents baseline.  
  
so now we have to modify the css for first three items like below and also add the baseline property in the container  
  
  
  
  
  
we get the below output in the browser like below.  
  
  
  
so if see above the text inside the each item is aligned across baseline and also we can change the flex direction to column and it will work the same.  
  
7) **align content**  
 This property aligns the line of content along the cross axis basically it is just short mix of justify   
 content and align items. It distributes additional space but along the cross axis. One important   
 condition is that multiple lines or multiple rows must exists within the container   
  
 so in the CSS file we will provide height and flex-wrap here wrapping is required to force our   
 content into at least two lines like below  
  
  
  
so we will get below output.  
  
As we can see if we reduce the browser width then content will wraps into two lines so the align content property is used to align these lines of content with in the container and the default value is stretch so if we add the align content property and set the value to stretch like below we still get the same output as for previous.  
  
  
  

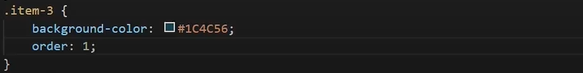
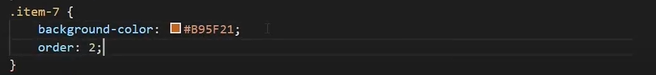
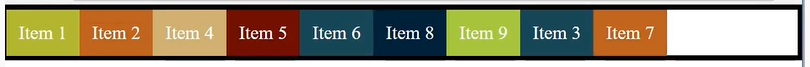
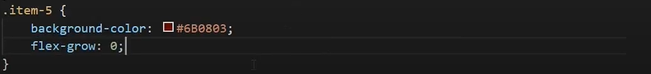

so second possible value is flex-start like below and it pulls the line to the beginning of the cross axis.  
  
  
  
so for the above we will get the below output and if we see the cross axis runs from top to bottom so the content of lines is pushed to the very top flex start i.e the start of the cross axis.   
  

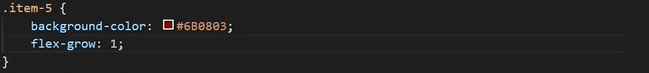
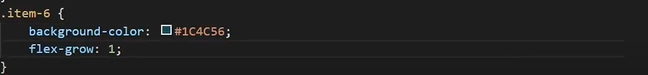

similarly, we also have flex-end which pushes the content to the end of the cross axis.  
  
  
  
For the above we will get below output. So the content is at the bottom.  
  
  
  
we can also centre the content by specifying the value to centre.

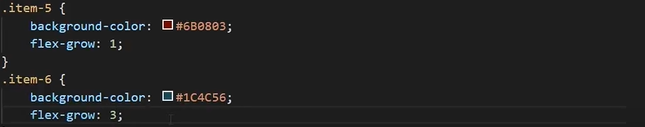
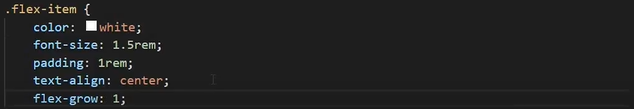
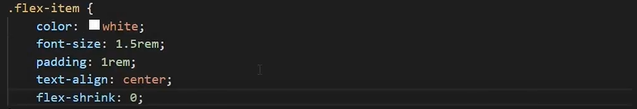
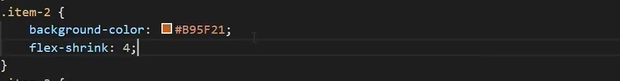
  
  
so, the above content will be centred.  
  
so the fourth value is space-between as this takes all the extra space and puts in between the lines.  
  
   
  
so we will get below output .

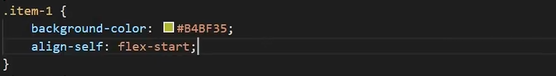
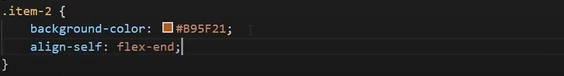
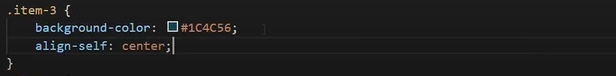
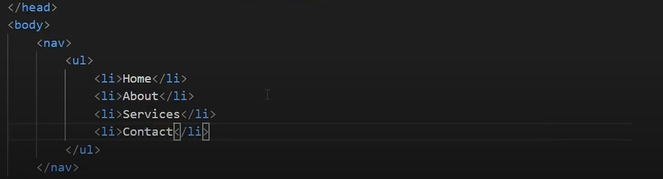
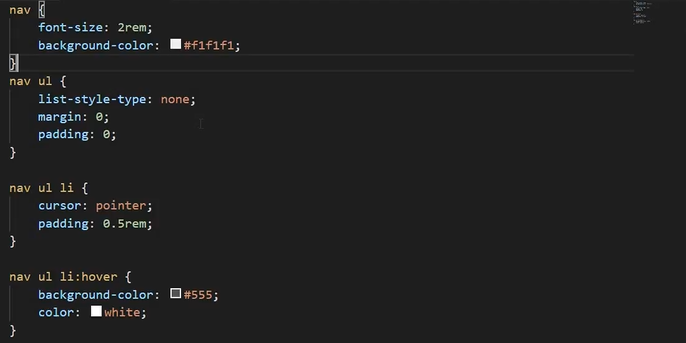
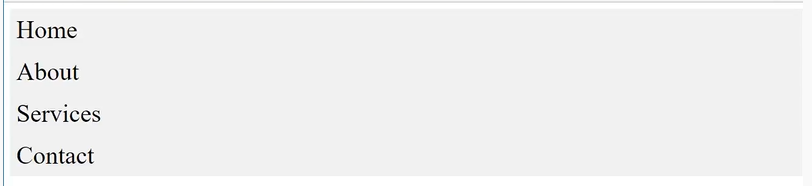
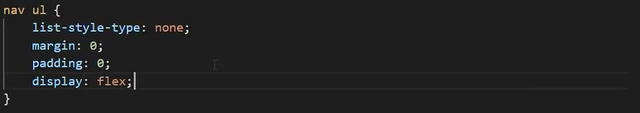
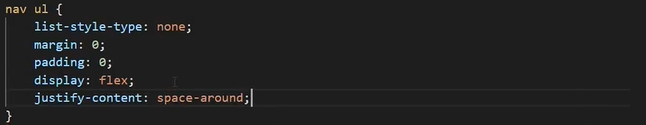
  
   
  
Finally, we have the space-around value which distribute the space around the lines like below.  
  
 

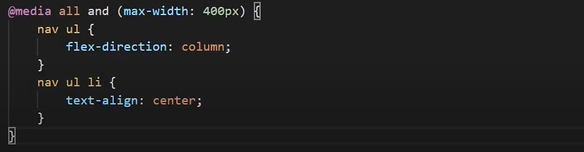
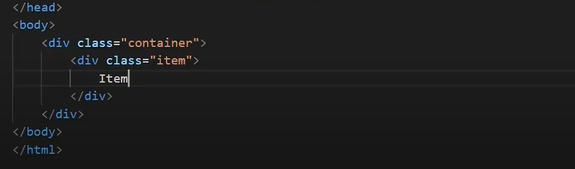
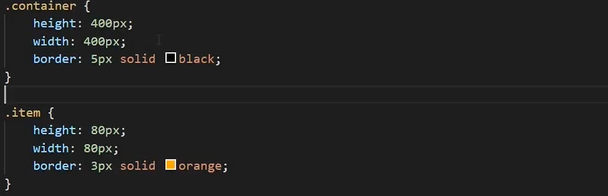
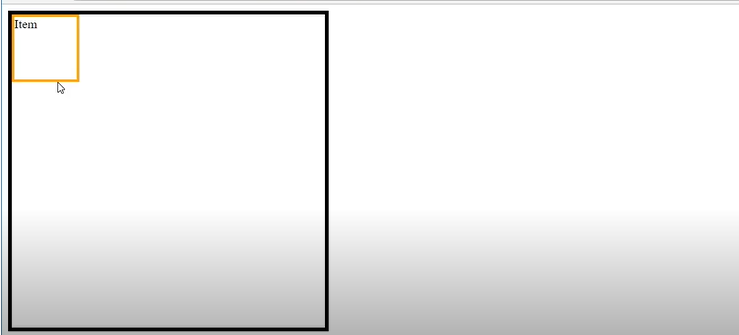
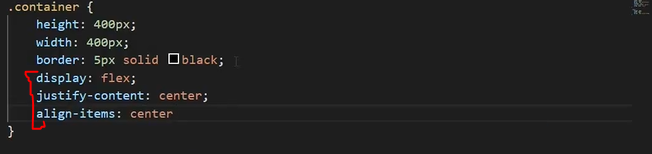
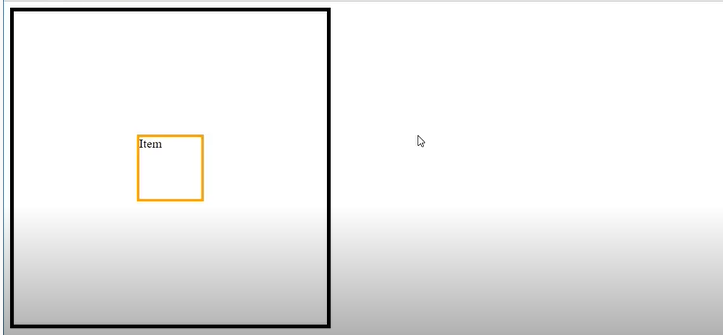
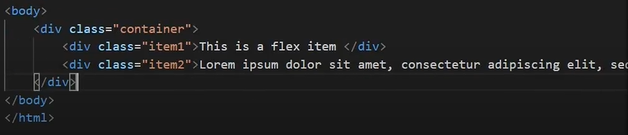
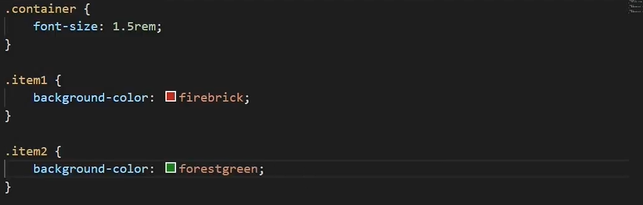
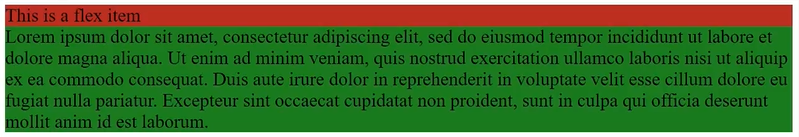
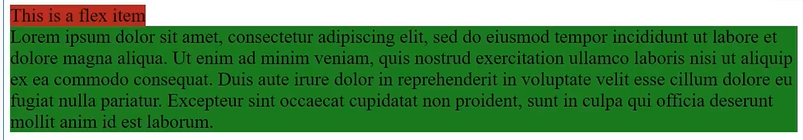
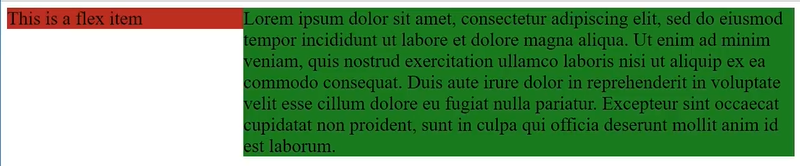
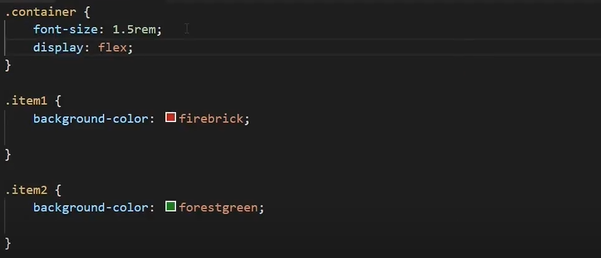
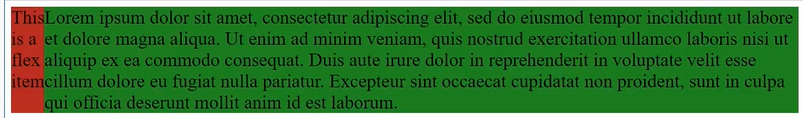
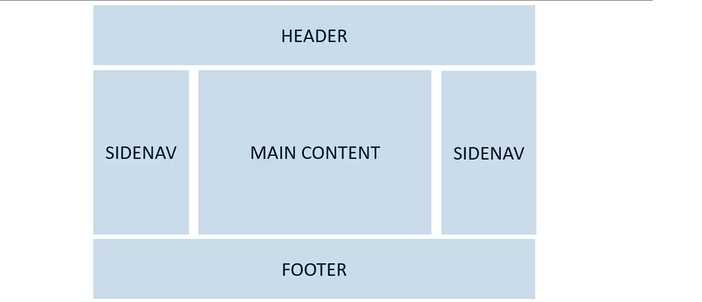
so, by using above property we have output like the centre portion has twice the space as the edges. So, if the space in between is 100 px then at the edges it will be 50 px.  
  
so far, we have learned the properties related to the container now we will learn related with the items.  
  
**Flex item properties**Below are the properties related to the Flex item  
  

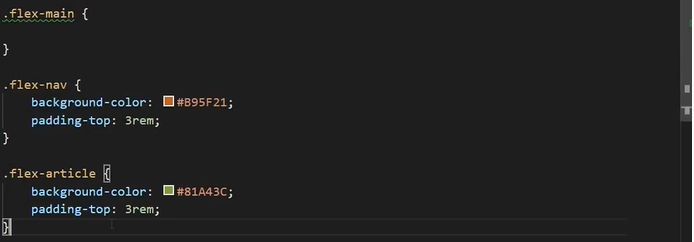
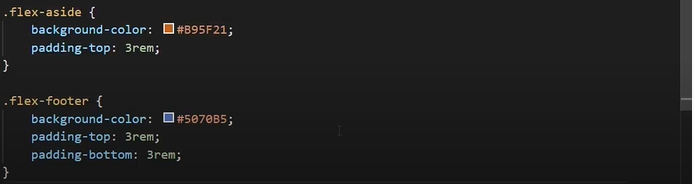
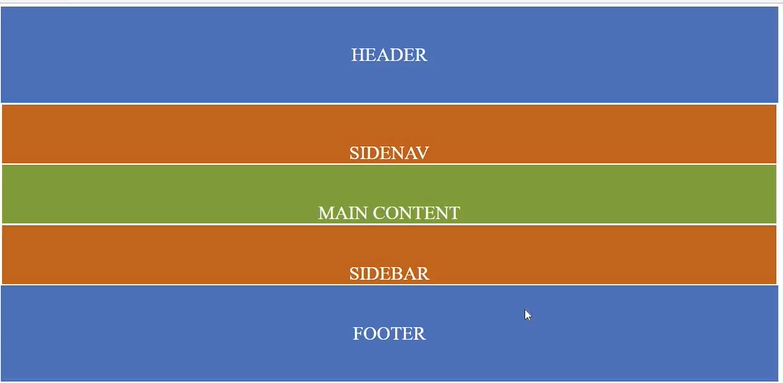
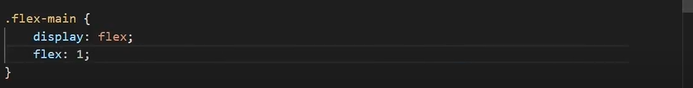
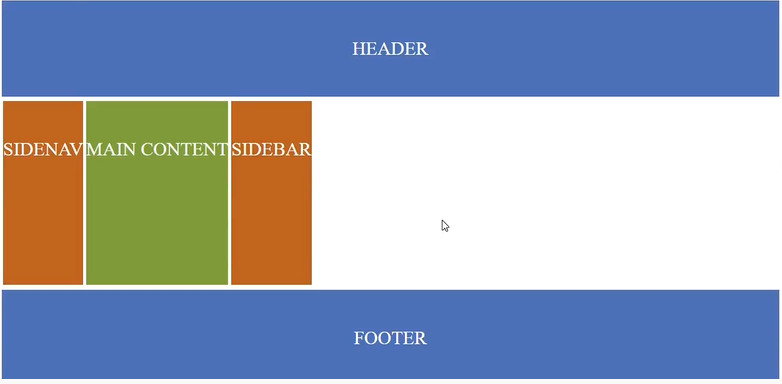
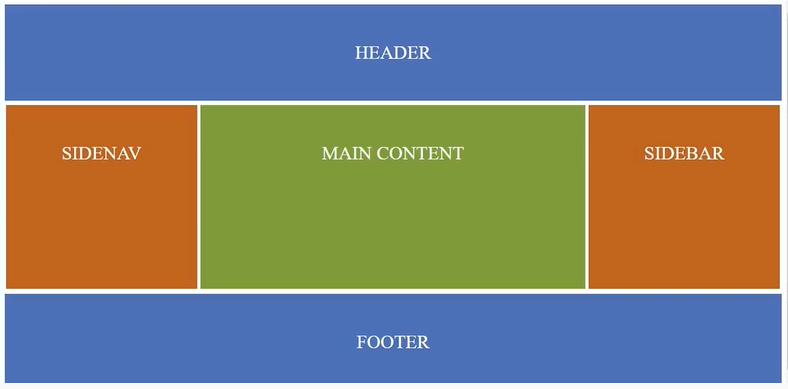
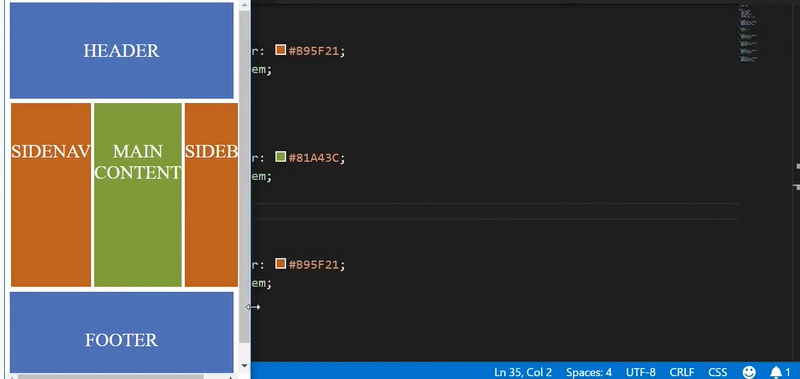

1) **order** This property control the order of items in the flex container and it has an integer value. Flex items in a container are lay out in the order in which they appear in the source code and this   
 order can be changed using the order property.  
  
Initially we have this in the browser and the items are appeared as they appeared in the source   
code like below.  
  
   
  
  
so now let us try to mix up the order so we make some changes in the CSS file like for the item 3 we have add the order of 1 like below in the code.  
  
  
  
so we get the below output in which item 3 is pushed all the way to the end and this is because all items by default have an order of 0.   
  
   
  
so once again we change the order by 2 of item 7 like below.  
  
  
  
so we will get the below output and we can see that item 7 is pushed to the very end like order 0 item come first then with order 2 then with order like so on  
  
  
  
but if you specify the same order no for more than one item ,items are laid out in the order in which they appear in the source code. So, if we specify order 1 to item 3 and item 9 but in the HTML file item 3 appears before item 9 so this is why we get the below output.  
  
  
2) **Flex grow**  
 This property specifies what amount of space inside the flex container the item should take up if   
 necessary. The flex grow factor is always relative to the size of the other items in the flex container  
  
By default the we can see that flex items only takes up place that is required to fit the content so there is lot extra space left with in the container.  
  
  
  
By default all flex items have a value of 0 that’s why they don’t take extra space available. So we have the change the flex-value to 0 for the item 5.  
  


But there will be no change in the browser output 0 is the default value. So when we change the value to 1 like below.  
  
  
  
we will get the below output as we can see that item 5 grows to take up the all the remaining space in the container.  
  
  
  
Now flex-grow value 1 to the item 6  
  
  
  
we get the below output as the additional space has been evenly distributed among the flex items 5 and 6 so if there is an 100 space left then both will occupy 50 px each.  
  


So, if we change the value to 3 for item 6 and item 5 remains 1 like below in the code.  
  
  
  
we get the below output as item 6 takes the three times extra space as the item 5 takes as flex- grow property works relatively and we can specify decimal values as well but do not specify negative values.  
  
  
  
so if we want to grow to take up the extra space evenly for that we need to add flex-grow and set it to 1 for all the flex items so comment the flex-grow of item 5 and item 6 and mention the flex-grow in the flex-item class like below.  
  
  
  
we get the below output all items grow evenly to take over the left space.  
  
  
  
3) **Flex shrink** It defines the ability for a flex item to shrink if necessary and default value is 1 and it is relative to   
 other items in the container.  
  
so if we try to reduce the width of the parent container then the flex items shrink to fit inside the container.  
  
  
  
and the shrinking is possible only up to certain point and if we reduce the more width then overflow will occur.so if we don’t want shrinking to be happened on the flex items for that simple set value to 0 for all the flex items.  
  
  
and we get the output if we reduce the width then there will be no shrinking but the item will overflow right away as item 9 is not visible.  
  
  
  
so will mention flex-shrink value to 4 for the item 2 and comment the other shrink values.  
  
  
  
so here item 2 is going to shrink 4 times as much as the other items as we can see the output if we reduce the browser width.  
  
    
  
  
4) **Flex basis**  
 It specifies the initial size of a flex item before the extra space is distributed in the container.  
 It can accept values in percentages, rems, ems, px and keyword like auto etc.  
 By default it has value of auto.  
   
 change in the html like below   
  
  
  
Now the item 1 has larger initial width  
  
  
  
Now changing the flex-basis property to 400 px  
  
  
  
we will get the below output.  
  


5) **Flex** Basically it is shorthand for flex grow, flex shrink and flex basis.  
 The flex property accepts three values separated by space flex-grow, flex-shrink and flex-basis   
 respectively.  
  
 By default flex property has an value of **0 1 auto**.  
  
 So , the flex property can be specified using either one two or three values please refer below   
 figure.  
  
   
  
6) **Align self** It is used to control alignment of individual flex items   
  
 so to use this first we need to mention height in flex container up to 400px. So currently we have   
 this output   
  
  
so first possible value for align-self is flex-start.  
  
  
  
we get the below output.  
  
  
  
second value is flex-end which align the items with the end of the cross axis.  
  
  
  
 we will get the below output for the above.  
  
  
  
Third possible value is centre.  
  
  
  
we will get the below output   
  
  
  
Fourth value is stretch and it stretches the item from cross start to cross end and also the default value is auto not the stretch.  
  
so remember one thing align-items on the container has the default value of stretch but align-self on the individual item has default value of auto and if align-self is provided it always overrides the align-items value of the flex container.  
  
  
 **Responsive Navbar**  
  
So we will design a responsive navbar basically a navbar is a common pattern for navigation where we have a list of items displayed as a horizontal bar.  
  
so we have two files html and css let us first take a part of the html file.  
  
  
  
and we have the css file like below.  
  
  
  
so for the above files we get the below output in the browser.  
  
  
  
so lets us convert into a navbar so to convert into a navbar we simply make unordered list tag as a flex container .  
  
  
  
so we can see the items are placed next to each other but we ended up with some extra space.  
  
  
  
so to deal with the extra space we can either add the space around the items or we add the space as a part of item itself i.e padding to the text  
  
so to distribute the space around the items we will use the property justify-content like below.  
  
  
  
we get the below output.  
  
  
  
  
so if we hover the menus you can see that it highlights only around the text content but if you want the spacing to be a part of the menu item itself we can use the flex-property on the individual item .   
  


so on each list items we can set flex and also the text to the centre and comment the justify-content property in the nav ul.  
  
  
  
so we get the below output.  
  
  
  
  
so we can add the media query to make it responsive and all we don in the media query is change the flex direction from row to column on the unordered list by doing this the menu items flow from top to bottom and we have set the max-width to 400 px so when we reduce the browser width less than 400 px then this properties will be applied.   
  
  
  
so, like below we get the output.  
  
  
  
  
  
  
  
**Centering an item**  
  
so we have below html and css file   
  
HTML file  
  
  
  
  
CSS File  
  
  
  
so initially we have this output in the browser we have item placed at the top left of the container.  
  
  
  
  
so we use the below properties in the container.  
  
  
  
so we get the below output.  
  
  
  
  
**Variable heights**  
  
consider a simple scenario placing two items side by side.  
  
HTML file it has two items one is item 1 which is only some short text and item 2 is along paragraph  
  
  
   
CSS file  
  
  
  
 if we see the browser output like below.  
  
  
  
if we could see second item is below item 1 but what we need is side by side for that we make use of float. So, after doing below changes in the CSS file. We get below output.  
  
  
  
still it is not right to fix this we need to give height to each item like below.  
  
  
  
we get the below output.  
  
  
  
so basically it’s a common issue that height is not the same for both the items of the container getting equally height on the children of a container. So, flexbox solves it easily.   
  
  
  
  
we get the below output  
  
  
  
Holy Grail Layout  
  
The holy fray layout is a page with a header, footer and 3 columns like below.  
  
  
  
so we are going to build the above page using flexbox.  
  
HTML file  
  
  


CSS file  
  
  
  
  
  
so after saving above files we will get the below output.  
  
  
  
but the main content we need in the horizontal wise so for that we make below changes in the CSS file.  
  
  
  
so we get the below output in the browser which is still sort what we want.  
  
  
  
But now hat we want is that the items takes up the whole space both horizontally and vertically.  
Now to grow the main content vertically we set the flex property to 1 like below.  
  
  
  
  
  
so to take up the horizontal space we need to add the flex property on the individual flex items sidenav, main content and sidebar so for both sidenav and sidebar add flex and set it to 1 1 and 5 rem so 5 rem sets the flex basis which is the initial width to begin with and for the main content we are going to add the flex property but with the different values as we want more content to take more of the additional space compared to the sidenavs so the main content will grow and shrink 3 times as quickly as the sidenavs like below.  
  
  
  
  
so finally we get the expected result.  
  
  
  
but if we reduce the browser width the layout does not looks good like below.  
  
  
  
so now we will make it responsive so for that we will add media query in the CSS file like below.  
  
  
  
so if we reduce the browser width than the 540 px then the above properties will take place.  
  
