Continue

Panels: So far we have mostly used the star width/height, which specifies that a row or a column should take up a certain percentage of the combined space. However, there are two other ways of specifying the width or height of a column or a row: Absolute units and the Auto width/height. Let's try creating a Grid where we mix these: Button 1 Button 2 with long text Button 3 In this example, the first button has a star width, the second one has its width set to Auto and the last one has a static width of 100 pixels.The result can be seen on the screenshot, where the second button only takes exactly the amount of space it needs to render its longer text, the third button takes exactly the 100 pixels it was promised and the first button, with the variable width, takes the rest.In a Grid where one or several columns (or rows) have a variable (star) width, they automatically get to share the width/height not already used by the columns/rows which uses an absolute or Auto width/height. This becomes more obvious when we resize the window: On the first screenshot, you will see that the Grid reserves the space for the last two buttons, even though it means that the first one doesn't get all the space it needs to render properly. On the second screenshot, you will see the last two buttons keeping the exact same amount of space, leaving the surplus space to the first button.This can be a very useful technique when designing a wide range of dialogs. For instance, consider a simple contact form where the user enters a name, an e-mail address and a comment. The first two fields will usually have a fixed height, while the last one might as well take up as much space as possible, leaving room to type a longer comment. In one of the next chapters, we will try building a contact form, using the grid and rows and columns of different heights and widths. This article has been fully translated into the following languages: Is your preferred language not on the list? Click here to help us translate this article into your language! For some reason every time I use a Grid control in WPF, I get these two settings mixed up.  The difference is subtle but simple, and it has a huge impact on how your content is displayed.  The important thing to remember is that â€˜Autoâ€™ will size the row or column based on its content, while â€˜*â€™ will size the row or column based on the size of the Grid. Setting the Height or Width to â€˜*â€™ Setting the height or width of a column or row to â€˜*â€™ will force the row or column to fill any â€œemptyâ€ space until the width or height of all the rows and columns is equal to the width or height of the grid.  For example, if you define a grid with the following rows: This will result in a second row that expands vertically to the height of the grid, minus the first row height.  If instead, both rows are given a height of â€˜*â€™: The rows will both expand vertically to fill the Grid.  Each subsequent â€˜*â€™ is sized so that the heights of all the rows exactly equal the height of the Grid, and each â€˜*â€™ height is equal to the others.  Finally, you can define how each height or width relates to the others by specifying a factor: In this case, the second row will be twice as tall as the first, subject to the same constraints as before. Setting the Height or Width to â€˜Autoâ€™ Setting a height or width to â€˜Autoâ€™ results in a row or column that will size itself based on its content.  If you defined the following columns: The second columnâ€™s width would be entirely based on the width of its contents.  If you had the following content: The first column would still have a width of 120, while the second column would only have a width of 80.  Notice that it does not matter how wide the Grid itself is.  â€˜Autoâ€™ will always size itself to fit the size of its content. Gurukultree is a knowledge-sharing site for tech geeks. Our objective is to share knowledge, idea, research, thoughts, expertise with the community. Our team is continuously doing an effort to provide the knowledgeable and best content. We provide the platform so that everyone can come and be a part of our knowledge sharing team. For any query or suggestion, You can write us [email protected] * and 'AUTO' are little tricky while working with the Grid layout container. In Grid, child elements are arranged in cells defined by rows and columns. RowDefinition is used to define the Rows and ColDefinition is used to define Columns. * and Auto is used to define Row's height and Column's width. We will not discuss here, how to use RowDefination and ColDefination. We will keep our focus on * and Auto. There is actually three-way to set the Rows height or Column width. doubleValue starSizing(*) Auto - or - -or- doubleValue In doubleValue, the value is expressed in pixel.  starValue It allows you to specify columns and rows in percentages, * is need to use after the value. Auto The size is determined by the size of the content object. DoubleValue It is straight-forward. if a double value is put in Height property of RowDefinition then it will be the height of the row in pixel. See below example how to use this. Example :  in this example height of the row is 25 pixel Start Value it is used when we need to define height or width in percentage. it is the default unit. If no value is mentioned the default sizing is 1*. example 1 -or- example 2 means 1st row is 10 times the size of column 2. example 3 In this example, first-row height is 40% and the second-row height is 60%. Auto In Auto, size is resolved by the size property of the content control. example 3 In this example, Grid has two rows, the second row has a button and is Auto so it takes the height of the button but the first row is * so it will stretch and occupy all remaining space available. So with the example it is clear, In Auto it take the size of content object.  Note As we have shown the example for RowDefination, same is true for ColDefination also. Blog Post: Understanding the Difference between 'Auto' and '*' when Setting Width/Height for a Grid ColumnAre you feeling puzzled by the terms 'Auto' and '*' when it comes to setting the width and height of a grid column?  Don't worry, you're not alone! Many developers struggle with understanding the differences between these two options.In this blog post, we will demystify the distinction between 'Auto' and '*' and provide you with easy solutions to overcome common issues.Auto - Flexible Width/HeightWhen we set the width or height of a grid column to 'Auto', we let the browser automatically calculate the appropriate size based on the content within that column. It means that the column's width or height will adjust itself dynamically as you add or remove content inside it. For example, imagine you have a grid column with the width set to 'Auto' containing a sentence. Initially, the column's width will be equal to the width of the sentence. However, if you add more text or increase the font size, the column's width will expand accordingly, ensuring all the content is visible without being cut off. Auto-sizing is excellent for responsive designs or situations where the content's size might vary dynamically. * (Asterisk) - Equal DistributionThe '*' symbol, on the other hand, represents equal distribution of available space among the grid columns. It means that each column will have an equal share of space, regardless of the content within it. Let's say you have a grid with three columns, each set to a width of '*'. In this case, each column will consume one-third of the available space, regardless of its content's size. This ensures that all columns are evenly spread out and take up an equal amount of space.The 'Auto' and '' options are not mutually exclusive. You can combine them to achieve specific layouts. For instance, you might want one column to dynamically adjust its width based on its content ('Auto'), while two other columns are equally distributed ('').Common Issues & Easy Solutions⚠ Problem: I am unable to align my grid columns, and they don't seem to consume the same amount of space. ✔ Solution: Ensure that you are applying the '*' width/height option to all the grid columns you want to be equally distributed.⚠ Problem: My grid columns are overlapping or overflowing the container. ✔ Solution: Try setting the width/height of your problematic columns to 'Auto'. This will let them adjust their sizes based on the content, preventing any overflow issues.⚠ Problem: The grid is not responsive on different screen sizes. ✔ Solution: Experiment with a combination of 'Auto' and '' options to create responsive grid layouts. Set widths/heights to 'Auto' for flexible columns and '' for those that need equal distribution.ConclusionUnderstanding the difference between 'Auto' and '' is essential when it comes to creating well-structured grid layouts. 'Auto' allows for flexible sizing, dynamically adjusting to content changes, while '' ensures equal distribution of available space among columns.Remember to experiment with different combinations of these options to achieve your desired layout, and don't hesitate to seek further resources on grid layouts and CSS techniques.We hope this guide has cleared up any confusion you had regarding 'Auto' and '*'. Now go forth and create stunning grid-based designs with confidence! 😎Have you encountered any other grid layout challenges or found unique solutions? Share your experiences in the comments below! Let's learn from each other and further enhance our grid game.  CALL-TO-ACTION: Join our community of web developers to explore more CSS tips and tricks to level up your skills! Sign up for our newsletter to receive regular updates that will keep you at the forefront of web design trends. Don't miss out – join now and become a grid ninja!  .Net Core Version History September 16 2024 .Net Core How to get parent control in wpf? March 23 2022 WPF November 07 2021 WPF Understanding Resources in WPF November 07 2021 WPF Introduction of Behaviors in WPF November 07 2021 WPF Read and Write in registry from C# October 17 2021 CSharp General Interview Question - Part1 September 14 2021 Interview Question July 25 2021 Xamarin September 05 2020 CSharp Transformations in WPF (2D Transformations) September 03 2020 WPF August 08 2020 WPF How to Convert a string to an enum in CSharp? July 26 2020 CSharp Make hyperlink open in microsoft edge July 24 2020 HTML How do I redirect to another webpage? July 01 2020 Jquery/Javascript ICommand Interface In MVVM - WPF May 02 2020 WPF How to inherit the style from default style in WPF April 27 2020 WPF Install and Environment Setup for Angular March 29 2020 Angular9 How to enumerate an enum in CSharp February 12 2020 CSharp December 28 2019 Angular9 Difference between * and Auto in wpf grid December 25 2019 WPF Implement Splash Screen in WPF December 21 2019 WPF Visual Tree and Logical Tree in WPF December 07 2019 WPF Different Types of Templates in WPF December 07 2019 WPF December 07 2019 WPF December 07 2019 CSS How to find the index of key in dictionary December 07 2019 CSharp Reddit and its partners use cookies and similar technologies to provide you with a better experience. By accepting all cookies, you agree to our use of cookies to deliver and maintain our services and site, improve the quality of Reddit, personalize Reddit content and advertising, and measure the effectiveness of advertising. By rejecting non-essential cookies, Reddit may still use certain cookies to ensure the proper functionality of our platform. For more information, please see our Cookie Notice and our Privacy Policy. Thanks for contacting us For any query, issue and information. You can write us [email protected] You can use the SharedSizeGroup attribute to share column sizes across different grids. You can’t normally share sizes of star-sized columns across multiple grids, however, since star sizing works only in the context of the current grid. Below is an example of how to get a star sized column to be the same size as a star sized column in a different grid. In the example, we have two grids. In the first, we have two auto-sized TextBlocks and then a star-sized TextBox that takes up the remaining space. In the second grid, we have just a TextBlock and a TextBox, but we want the TextBox to be the same size as the one in the first grid. Using SharedSizeGroup on the TextBox columns won’t work, since the columns would then get auto-sized. The solution is to set up a SharedSizeGroup on every column except for the TextBox columns and to add a dummy third column in the second grid. The TextBox columns will then each independently use star sizing and end up the same size. If one or both columns on either size of a GridSplitter (in its own column) have a width set to Auto, the behavior is as follows (“proportional” refers to star sizing): Both columns set to Auto: Left column changes size and is switched to Absolute sizing Left column Auto, right column proportional: Both columns change size, left switches to Absolute, right remains at original proportion (e.g. “1*”) Right column Auto, left column proportional: Both columns change size, right switches to Absolute, left remains at original proportion In the example below, columns 0 and 2 surround a GridSplitter in column 1.  Columns 0 and 2 are set to Auto size (as is the column with the GridSplitter) and column 3 is set to proportional. Initially, we see that both columns 0 and 2 are auto-sized. If we slide the GridSplitter to the right, column 0 changes size and switches to Absolute sizing.  Column 2 remains as Auto. Recall that a GridSplitter in its own column and with its HorizontalAlignment set to Center will resize columns on either side of it.  In the example below, columns 0 and 2 are resized, but the width of column 3 is not changed. If all of the content columns are sized using star sizing, the GridSplitter will leave them as star sized, but change the coefficients to match the final size after moving the GridSplitter.  We can see this by dumping out the column widths before and after our sizing operation. Before resizing, columns 0, 2 and 3 are all “1*” (or just “*”).  They take up equal space. After resizing, the coefficients change.  Columns are still proportionally sized, but the proportions are different. We can verify that the columns are sized proportionally by resizing the entire Grid (containing window).  The columns retain their post-GridSplitter relative sizes. You can use star sizing when setting the height of rows or width of columns in a Grid to distribute space in a Grid across multiple rows or columns.  You can use integer values preceding each ‘*’ (star) to indicate the relative size of rows or columns. You can also use floating point values as relative star sizing numbers.  In the example below, we use numbers between 0.0 and 1.0 and set them so that their total adds up to 1.0.  This allows using the values as a ratio of the available space. You can use star sizing when setting the height of rows or width of columns in a Grid to distribute space in a Grid across multiple rows or columns. Here’s an example. The space allocated for each star-sized row is calculated as: n / sum, where n = number preceding * for that row; sum = sum of all numbers preceding star-sized rows Row is given n/sum of the total available space, after allocating space for Auto and Absolute sized rows You can use star sizing when setting the height of rows or width of columns in a Grid to distribute space in a Grid across multiple rows or columns. When you specify the size of a row or column with star sizing, you can specify the size simply as “*”, or you can specify a number followed by the star (e.g. 2*).  The number indicates the size of the row or column, relative to the other rows or columns.  (The value “*” is equivalent to “1*”). In the example below, the second row’s height is specified as “2*”, indicating that it should always be twice the height of the first row.  (Or 2 / (1+2) = 2/3 of the total) You can use star sizing when setting the height of a row in a Grid or the width of a column.  After a Grid allocates space for rows and columns that are sized using absolute values or auto-sized to fit the row/column contents, it will distribute the remaining space in the Grid across rows and columns that use star sizing. In the simplest case, if the height or width is specified as “*”, space is distributed evenly across all remaining rows/columns. Star sizing is the default for a row’s height or a column’s width, if you do not explicitly set a height or width.