Grid Layout – Part 1

Grid is a layout system that uses rows and columns and is easier to layout your page without having to use floats or positioning. Grid is the value of the display property. When you have an element with children you can apply the display: grid declaration to the parent (or container) and set up the columns, and the children will become the grid items and be placed into the grid. Once the columns are used up on one row, another row will be created. So the grid consists of a parent element and one or more child elements.

Initially the children will be placed into the grid, as grid items, in the order that they were in HTML, but later we will see that we can place the children anywhere in the grid that we want to.

Let’s quickly define explicit and implicit. When the browser automatically placed the grid items or children in the order, they were in the spaces available automatically this is implicit. Explicit is when we as the developer define it ourselves. We use a property that set something up or places something where we want it. So explicit is we do it deliberately and implicit is when it just happens automatically.

Let’s look at a simple grid example.

If I had a parent element with 10 direct children. It would layout like this in normal flow. Divs are block level and they take all the width each one on its own line going down the page. If I used the parent element as the selector in CSS and gave it the property of display: grid. It is now ready to be a grid. Nothing changed because we have to set up the columns for the grid. The property for that is grid-template-columns and however many length values we give this property will show how many columns we will have. I could give it 3 columns of any length like 200px 200px 200px. Notice that I didn’t set up rows but the children fill up the 3 columns and then wrap around to the next row automatically when the row above is filled. The rows are created implicitly. There is a grid-template-rows property if you want to explicitly set those up, but I rarely use the grid-template-rows because the children will implicitly go into new rows as needed. But I do explicitly always set up the grid-template-columns.

We can also give a gap to the areas between our columns and rows with the grid-gap property. I could give that area 30px and now the lines between the columns and rows will grow by that size.

Let’s try 4 columns with a 4th value. Let’s give differing length this time as well. Something like 200px 70px 100px 20px; Now I have 4 columns of different widths. You can use other measurements like em or auto as well. Instead of percentages, though, we will use a new measurement called fr (more about fr in a minute). Let’s look at auto. Let’s try 2 columns one of 150px and one with auto. Auto will automatically take whatever space is left after the 150px. It’s great for responsiveness. If I change the width of my screen, you can see it takes whatever space it can. Fr is also a great responsive unit of measurement with grid. Fr is the fractional unit of measurement. It works a little like percentages but better because with percentages, it’s easy to cause our grid to go too wide with grid gaps and the user would have to scroll horizontally. With fr it’s only going to take what’s available and won’t go over that amount. The unit measurement such as 1fr gives us 1 fraction of the whole that is available. If I were to set up my column as 1fr 1fr 1fr. Then, I would have 3 equal fractions of the whole. Or in other words 1/3, 1/3 and 1/3 of the space available. It is also responsive as well as I change the screen size, which is really nice.

I can again change the number of columns to 5 if I wanted with 1fr 1fr 1fr 1fr 1fr and now I have five columns of 1/5th the available space. When you start to have the same measurement repeated this many times, there is a shortcut you can use that will do the same thing with repeat(5, 1fr). Either way it would give me 5 columns of equal widths. If we wanted one column wider, we simply could make one for example twice as big as the others. If we had 1fr 2fr 1fr. The middle column is taking twice as much space, 2 fractions of the whole. So, we have a total of 4 parts of the whole. Our columns are taking 1 part, then 2 parts and then one part. ¼, ½, ¼ .

We can also nest grids. We can have a grid inside of another grid. One of the children of the grid could also be a grid container or parent of another grid if it had children.

More on explicitly setting grid items or children into specific areas of the grid later.