Grid Layout - Part 1

Female: Grid layout, part one. Grid is a layout system that uses rows and columns. *[A blue square is on the screen with lines over it to create a grid.]* And it's easier to lay out 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 then set up the columns. And the children will become the grid items and be placed into the grid.*[The boxes of the grid are all labeled with a number one through seven, except the bottom two which are left empty.]* Once the columns are used up on one row, another row will be created. So the grid consists of the 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're going to see that we can place the children anywhere in the grid that we want to. Let's quickly define explicit and implicit.*[The words "Explicit" and "Implicit" show up on the screen.]* When the browser automatically placed the grid items or children in the order they were in the spaces available automatically. This is implicit.*[The word "Automatically" shows up next to the word "Implicit".]*

Explicit is when we, as the developer, define it ourselves.*[The word "Manually" shows up next to the word "Explicit".]* We might use a property that set something up or places something where we want it to. So explicitly, we will do it deliberately, and implicitly is when it just happens automatically.

[The text fades away and a new visual is shown. On the left side is a tab showing a grid with one column, with each row labeled a number one through ten. On the right side is a tab showing HTML text.]

Let's look at a simple grid example. If I had a parent element with ten direct children, it with layout like this in normal flow, divs or block level and they take all the width they can. So each one is on its own line going down the page. *[On the right side, she types "display: grid;"]* If I use 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. *[In the next line she types "grid-template-columns: 200px 200px 200px;"]* The property for that is grid template columns. And however many length values we give this property. It will show that many columns, I could give it three columns of any length by 200 pixel, 200 pixel, 200 pixel. *[The grid becomes three columns.]* Notice that I didn't set up rows, but the children fill up the three 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 up those. But I rarely will use the grid template rows because the children will implicitly go into new rows as needed. But I do always explicitly set up the grid template columns.

We can also give a grid gap to the areas between our columns and rows. *[Types "grip-gap: 30px;" and the grid expands so the boxes are not touching.]* With the grid gap property, I could give that area 30 pixels. And now the lines between the columns and rows will grow by that size. Let's try four columns with a fourth value. Let's give differing links this time as well. Something like 200 pixels, 70 pixel, 100 pixel, 20 pixel.

[Next to "grid-template-columns:" she types "200px 70px 100px 20px." Each column of the grid changes size.]

Now I have four columns of different widths. You can use other measurements like EM or auto as well. But instead of percentages, we're going to use a new measurement called FR. More about FR in a minute. Let's look at auto. Let's try two columns, one of 150 pixels and one with auto.

[She deletes text next to "grid-template- columns:" and replaces it with "150px auto;". The grid size changes.]

Auto will automatically take whatever space is left after the 150 pixels. 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 percentage, but better. Because with percentages, it's easy to cause our grid to go too wide. Especially if we have a grid gap than the user would have to scroll horizontally. With FR or fractional units. It's only going to take what's available and it won't go over that amount. *[Replaces "150px auto;" with "1fr 1fr 1fr;". The grid changes to three equal*

columns.] The unit measurement, such as a one FR, gives us a 1 fraction of the whole that is available. If I were to set up my cons as one FR, one FR, when FR, then I would have three equal fractions of the whole. Or in other words, 1 third, 1 third, 1 third of the space available. It's also responsive as well. As I change the screen size, which is really nice. I can again change the number of columns to five if I wanted, with a one FR, when FR, FR, FR, when FR. *[Adds "1fr" twice. This creates five columns on the grid.]* And now I have five columns of 1 fifth of the space available. When he started to have the same measurement repeated this many times. There is a shortcut you can use that will do the same thing. Repeat five comma one FR. Either way, it's going to give me five columns of equal width. *[Replaces the five "1fr"s with "repeat(5, 1fr);". The grid does not change.]* If I wanted one of the columns to be wider, we simply could make one, for example, twice as big as the others. If we had one FR, two FR, one FR. *[Replaces "repeat(5, 1fr);" with "1fr 2fr 1fr;". The middle column gets wider.]* The middle column is taking twice as much space to fractions of the hole. So we have a total of four parts of the whole. Our columns are taking one part then two parts, and then one part, 1.5 fourth, 11 fourth. We can also nest grids. We can have a grid inside of another grid. What if one of the children of the grid, it could also be a container or a parent of another grid if it had children. More on explicitly setting grid items or children into specific areas of the grid later.

[End of video]