**Grid Gallery Demo**

Start with the base files of index.html and style.css. Explain how things are configured – it’s just a bunch of divs each with some html content (text) – its image is being set in the CSS via the background-image property. We do this so that we can directly manipulate the image more easily with CSS. We also have a bunch of CSS styling each “card” but no need to worry too much about that

The container <section class=”photo-grid”> will be the grid and All of the <div class=”card”> elements inside are grid items

This doesn’t look good right now – because we have no grid set up. So lets do that.

We just set display:grid on the container element. We can also use gap like we did with flexbox

Next, we just have to specify how wide we’d like each column to be. We can manually specify a width for each column like in style2.css

grid-template-columns: 250px 250px 250px 250px 250px 250px 250px 250px 250px 250px 250px 250

but this is really ugly and not responsive. I can instead use something called a fractional unit, which basically just divides up however much space there is in the grid column.

I can also use the helper function repeat() to save a bit of time

.photo-grid {

display: grid;

gap: 1rem;

grid-template-columns: repeat(12, 1fr);

}

See style3.css - But this puts everything in one row and is ugly. We can use the minmax() function in a similar way to the clamp() function to set a minimum value

.photo-grid {

display: grid;

gap: 1rem;

grid-template-columns: repeat(12, minmax(250px, 1fr));

}

See style4.css. This makes each column a min of 250px, and will try to scale up to as large as 1 fr

But this overflows the page. Plus it assumes there will always be 12 items, which may not be the case. So we can use an implicit grid instead to let things break down

This is easy – instead of telling repeat() to repeat 12 times, we use “auto-fit” which tells the browser to figure out the number of columns + rows on the fly. ***Only 3 lines of code*** - style5.css

.photo-grid {

display: grid;

gap: 1rem;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

}

Okay so now we just need to work on the sizing of the individual cards. They’re really short because they are sized to the content of the div – which is the text. Remember, the image is just a background

One option is to specify how tall each row should be using grid-auto-rows

.photo-grid {

display: grid;

gap: 1rem;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

grid-auto-rows: 250px;

}

Style6.css

This is pretty good, but maybe we want some images to be taller or wider than others

First, we’ll make some additional classes - .card-tall and .card-wide

We want *tall* images to span across two rows

We want *wide* images to span across two columns

To do this, we use the grid-row property

grid-row: span 2 / auto

means we want it to take up 2 rows from wherever its starting position is

we use the slash to specify its ending point, which we can just let happen automatically

grid-row means start here / end here

we could specify *exact* rows and columns if we wanted, such as grid-row: 1 / 4

grid-column works basically the same way

.photo-grid {

display: grid;

gap: 1rem;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

grid-auto-rows: 250px;

}

.card-tall {

grid-row: span 2 / auto;

}

.card-wide {

grid-column: span 2 / auto;

}

See style7.css and then index2.html to see these classes applied to some of the images

But we’re left with one *small* problem, which is that on very small screens some some of the images still try to span 2 grid lines and it just doesn’t work well

We can fix this by wrapping our card-tall and card-wide classes in a media query – so that they are ignored on small screen sizes.

.photo-grid {

display: grid;

gap: 1rem;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

grid-auto-rows: 250px;

}

@media screen and (min-width: 600px) {

.card-tall {

grid-row: span 2 / auto;

}

.card-wide {

grid-column: span 2 / auto;

}

}

See style8.css and index2.html