Use the align-self property:-

This property allows you to adjust each item's alignment individually, instead of setting them all at once. This is useful since other common adjustment techniques using the CSS properties float, clear and vertical-align do not work on flex items.
'align-self' accepts the same values as 'align-items' and will override any set by the 'align-items' property.

# CSS GRID

Turn any HTML element into a grid container by setting its display property to grid. This gives you the ability to use all other properties associated with CSS grid.

```
display: grid;
```

Add columns with `grid-template-columns`

```
grid-template-column: 50pt 50px;
```

The number of parameters given to the grid-template-columns property indicates the number of columns in the grid, and the value of each parameter indicates their respective widths.

Add rows with 'grid-template-rows'

Similar to grid-template-columns.

## Use CSS Grid Units to change the size of column & rows

You can also use units like px & em but u can also use these
'fr': sets the column or row to a fraction of the available space.
'auto': sets the " " " " the width or height
of the content automatically
'%': adjusts the column or row to the percent width of its container.

Ex:- grid-template-columns: auto 50px 10% 2fr 1fr;

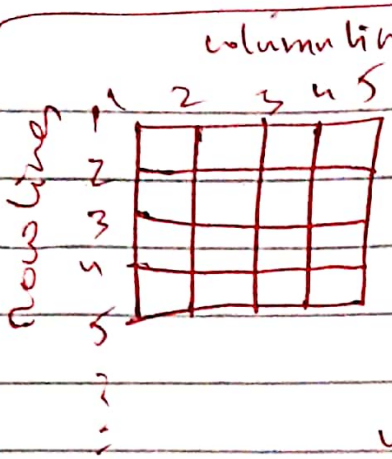
## Use grid-column-gap to create a Column Gap.

grid-column-gap: 10px;
creates 10px of empty space b/w all of the columns

To create a row gap,
grid-row-gap: 5px;

'grid-gap' property is a shorthand property for grid-row-gap and grid-column-gap. If it has 1 value it will create a gap between all rows & columns.
If it has 2 values, it will use the first one for rows and second for columns.
grid-gap: 10px 20px;

## Use 'grid-column' to control spacing:-




To control the amount of columns an item will consume you can do it with line numbers.

```
grid-column: 1/3;
```

will make the item start at the first vertical line of the grid on the left and span to the 3rd line of the grid consuming two columns.

Similarly 'grid-row' does for rows.

## Align an item horizontally using 'justify-self'

In CSS grid, the content of each item is located in a box which is referred to as a cell. You can align the content's position within its cell horizontally using 'justify-self'

```
justify-self: start (left of the cell)
justify-self: center
justify-self: end (right of the cell)
justify-self: stretch (fill the whole cell)
```

Similarly for vertical we have 'align-self'

For all items horizontally we have 'justify-items'
For all items vertically we have 'align-items'.

## Divide the Grid Into an Area Template.

You can group cells of your grid together into an area and give the area a custom name.

```
grid-template-areas:
    "header header header"
    "advert content content"
    "footer footer footer";
```

The code above merges the top three cells together into an area named 'header', the bottom three cells into a 'footer' area, and it makes two areas in the middle row; 'advert' and 'content'. Note:- Every word in the code represents a cell and every pair of quotation marks represents a row. In addition to custom labels, you can use a (.) period to denote an empty cell in the grid.

## Place items in Grid Areas using the 'grid-area' property:-

```
.item1 {
    grid-area: header;
}
```

This lets the grid know that you want the item1 element to go in the area named 'header'. In this case, the item will use the entire top row because that whole row is named as the header area.

## Use 'grid-area' property without creating an areas template

If your grid doesn't have an areas template to reference, you can create an area on the fly for an item to be placed like this

```
item1{
    grid-area: 1/1/2/4;
}
```

The above represents,

grid-area: horizontal line to start at/vertical line to start at/horizontal line to end at/ vertical line to end at;

## Reduce repetition using the 'repeat' function

To create a 100 row grid, each row 50px tall we can say,

grid-template-rows: repeat(100, 50px);

instead of writing 50px 100 times.

ex:- grid-template-columns: repeat(2, 1fr 50px) 20px;

is nothing but,

grid-template-columns: 1fr 50px 1fr 50px 20px;

## Limit item size using the 'minmax' function

Can be used with 'grid-template-columns' & 'grid-template-rows'

It is used to limit the size of items when the grid container changes size. We have to specify acceptable size range for the item.

Ex:- grid-template-columns: 100px minmax(50px, 200px);

this creates two columns, the first is 100px wide and the second has the min-width of 50px and max-width of 200px.

## Create flexible layouts using auto-fill

The 'repeat' function comes with an option called 'auto-fill'. This allows you to automatically insert as many rows or columns of your desired size as possible depending on the size of the container. You can create flexible layouts when combining 'auto-fill' with 'minmax'.

repeat(auto-fill, minmax(60px, 1fr));

When the container changes size, this setup keeps inserting 60px columns and stretching them until it can insert another one. Note: if your container can't fit all your items one row, it will move them down to a new one

'auto-fit' works almost same. The only difference is that when the container's size exceeds the size of the items combined, auto-fit collapses those empty rows or columns and stretches

your items to fit the size of the container.

- You can also create grids within grids.