ADVANCED GRID PROPERTIES

A. Repeat

`repeat()` is function (available to grid-template properties) to define a rows/columns.

Example:

```
.grid-container {
  grid-template-rows: 150px 150px;
  grid-template-columns: 150px 150px 150px 150px 150px;
```

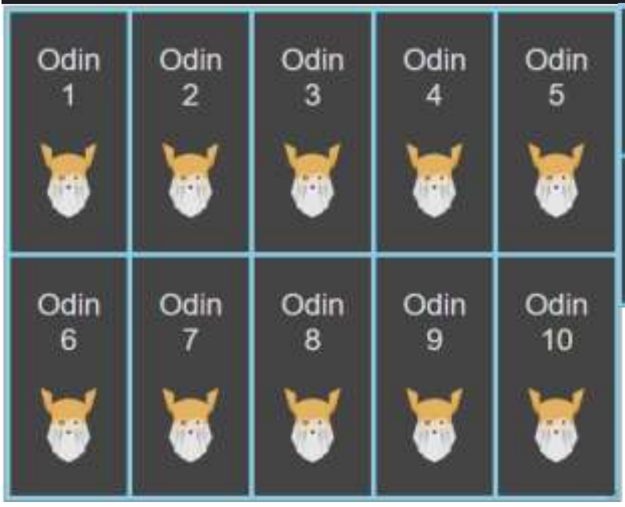
if we use repeat it'll become:

```
.grid-container {
  grid-template-rows: repeat(2, 150px);
  grid-template-columns: repeat(5, 150px);
```

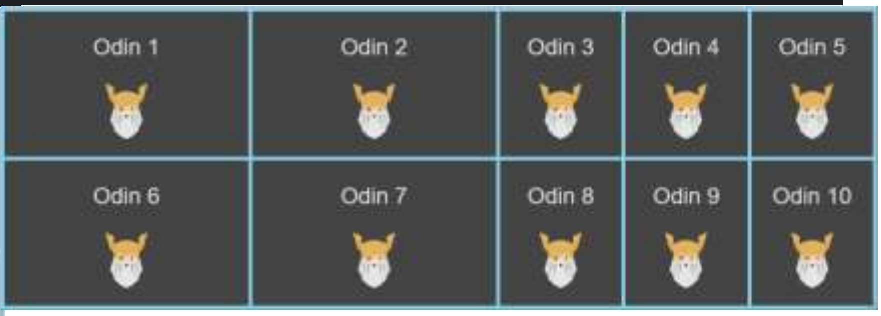
B. Fractional Units (`fr`)

The 'fr' unit is a way of distributing whatever remaining space is left in the grid. For example, if we have a four-column grid with a total width of 400px and four grid items each on a column track assigned 1fr as their size, all of the grid items should be given one fraction of that 400px of space, which is 100 pixels.

```
grid-template-rows: repeat(2, 1fr);
grid-template-columns: repeat(5, 1fr);
```



```
grid-template-rows: repeat(2, 1fr);
grid-template-columns: repeat(2, 2fr) repeat(3, 1fr);
```



Ketika container diperkecil sampai melebihi `min-content` value, maka grid-item akan tepotong.

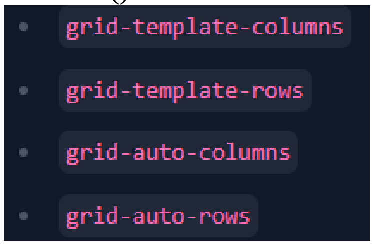
C. Minimum and maximum track sizes: min() and max()

```
.grid-container {
  grid-template-rows: repeat(2, min(200px, 50%));
  grid-template-columns: repeat(5, max(120px, 15%));
```

In this case, the grid row size will be calculated based on the smaller of the two values 200px and 50% grid container's height. Conversely, the grid column size will be calculated based on the larger of the two values 120px and 15% of the grid container's width.
In doing so, we are essentially setting a maximum height at 200px & minimum width of our grid column size at 120px.
Note: in min()/max() we should take 1 static argument & 1 relative argument (like px and %).

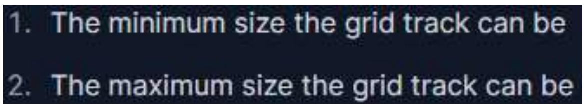
D. Dynamic minimum and maximum sizes

minmax() is a CSS function that is specifically used with Grid. It can only be used in:



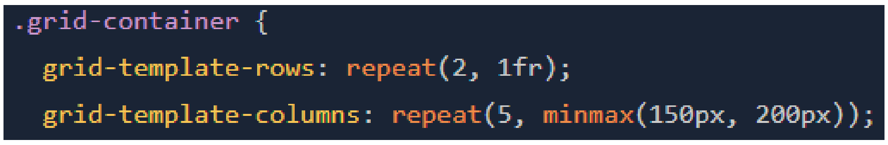
- `grid-template-columns`
- `grid-template-rows`
- `grid-auto-columns`
- `grid-auto-rows`

It is a relatively straightforward function that only takes in two arguments:



1. The minimum size the grid track can be
2. The maximum size the grid track can be

In minmax(), we can use use static values for both arguments.

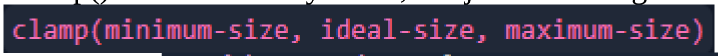


```
.grid-container {
  grid-template-rows: repeat(2, 1fr);
  grid-template-columns: repeat(5, minmax(150px, 200px));
```

The colums will have minimum height 150px, maximum height 200px, and will resize between it.

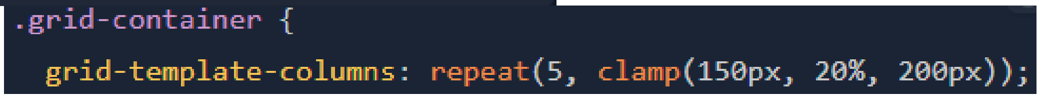
E. Clamp()

Clamp() can be used anywhere, not just within a grid container. Syntax:



```
clamp(minimum-size, ideal-size, maximum-size)
```

Example:

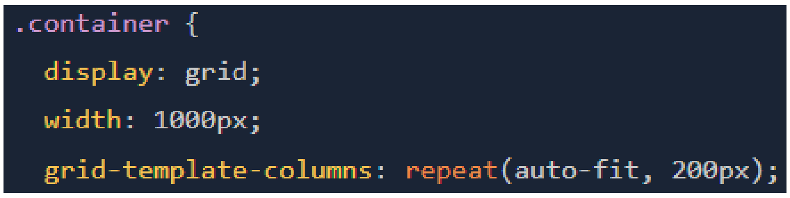


```
.grid-container {
  grid-template-columns: repeat(5, clamp(150px, 20%, 200px));
```

It willll render with a width equal to 20% of its parent's width, unless that number is lower than 150px or higher than 200px, in which case it will use those numbers as its width.

F. auto-fit & auto-fill

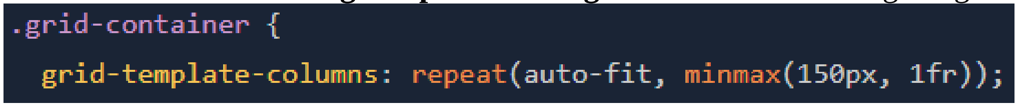
These two values are actually a part of the 'repeat()'. It's useful (to use with minmax()) to give your grid a number of columns that is flexible based on the size of the grid. For example, if our grid is only 200px wide, we may only want one column. If it's 400px wide, we may want two, and so on. Example:



```
.container {
  display: grid;
  width: 1000px;
  grid-template-columns: repeat(auto-fit, 200px);
```

For this grid, we have a set width of 1000px and we are telling it to fill in our columns with tracks of 200px each. This will result in a 5-column layout. In this case, 'auto-fill' would actually do the same thing.

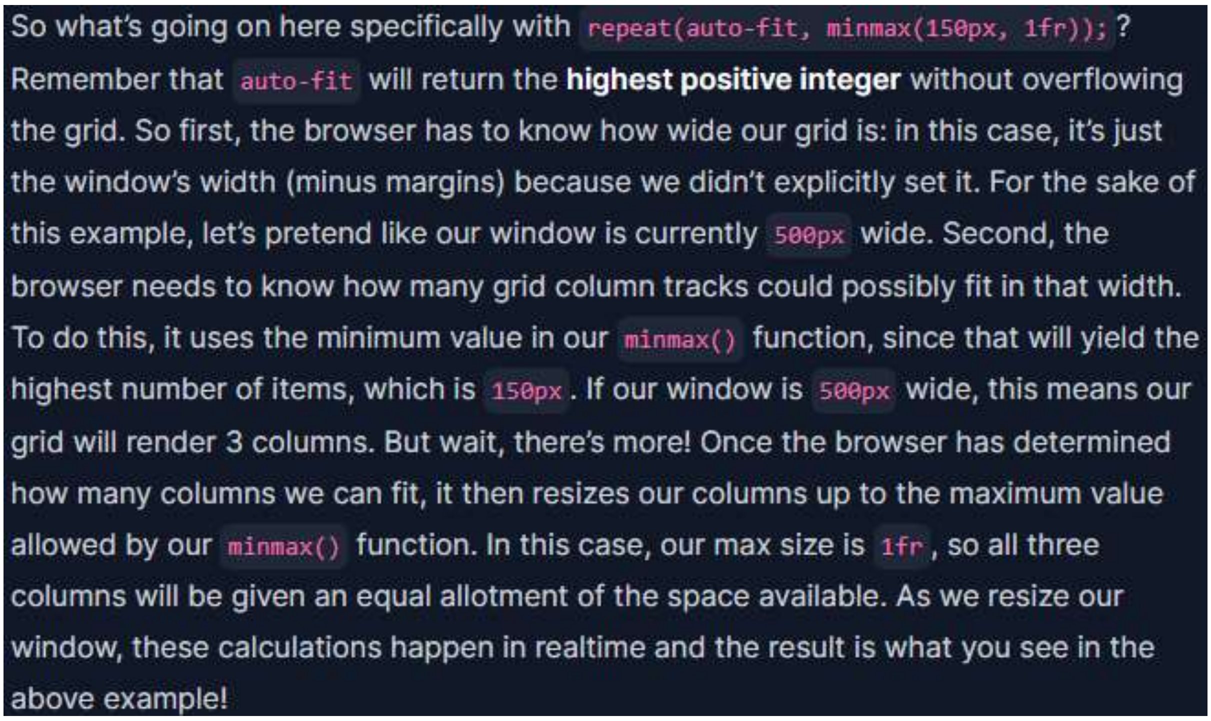
➢ auto-fit : will return the **highest positive integer** without overflowing the grid. Example:



```
.grid-container {
  grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
```

Penjelasan(di page selanjutnya, baru baca ini =>): jadi ketika container memiliki width 500px, masing2 grid-item akan memiliki width (nilai terkecil dari minmx yaitu) 150px sehingga terdapat 3 kolom/baris & terisi 450px. Berhubung masih ada ruang kosong 50px(500-450=50), maka grid-item akan memiliki width (nilai terbesar dari minmax) yaitu 1fr, dengan tetap mempertahankan 3kolom/baris sehingga terisi 500px(full width dari kontainer). Jadi jika lebar container itu kelipatan 150px, maka (masing2) grid-item akan memiliki width 150px. Jika lebar kontainer bukan kelipatan 150px, maka grid-item akan memiliki width 1fr.

With minmax(), we can tell our grid that we want to have as many columns as possible.


So what's going on here specifically with `repeat(auto-fit, minmax(150px, 1fr));`? Remember that `auto-fit` will return the **highest positive integer** without overflowing the grid. So first, the browser has to know how wide our grid is: in this case, it's just the window's width (minus margins) because we didn't explicitly set it. For the sake of this example, let's pretend like our window is currently `500px` wide. Second, the browser needs to know how many grid column tracks could possibly fit in that width. To do this, it uses the minimum value in our `minmax()` function, since that will yield the highest number of items, which is `150px`. If our window is `500px` wide, this means our grid will render 3 columns. But wait, there's more! Once the browser has determined how many columns we can fit, it then resizes our columns up to the maximum value allowed by our `minmax()` function. In this case, our max size is `1fr`, so all three columns will be given an equal allotment of the space available. As we resize our window, these calculations happen in realtime and the result is what you see in the above example!

Simulasi(ada di odin web): https://codepen.io/TheOdinProjectExamples/pen/abLzzgR

- auto-fill

  In most cases, 'auto-fill' is going to work exactly the same way as 'auto-fit'w. The difference is only noticeable when there are fewer items than can fill up the entirety of the grid row once.

  Ketika container diperlebar ke ukuran dimana muat 1 grid-item, tetapi tidak ada grid-item lagi, 'auto-fit' akan tetap membuat grid-tem menjadi ukuran max nya. Tetapi 'auto-fill' akan membuat grid-item menjadi ukuran min nya dan akan meninggalkan ruang kosong. Hal ini akan berlanjut terus seiring dengan bertambahnya lebar container.

  Misal: ada 3 grid item memiliki lebar masing2 100px & container memiliki lebar 350px. Maka 50px(350-300=50) akan dibagi ke masing2 grid item sehingga container terisi penuh. Tetapi ketika container diperlebar ke ukuran 400px maka 100px(400-300=100) itu akan kosong(karena muat untuk diisi 1 grid-item, sedangkan tidak ada grid-item lagi.

  Simulasi(ada di odin web):

  - auto-fit: https://codepen.io/TheOdinProjectExamples/pen/mdByJyJ
  - auto-fill: https://codepen.io/TheOdinProjectExamples/pen/KKXwpwX