**DOCUMENT OF GRID**

grid vs flex:

The main difference between CSS grid and CSS flexbox is the number of dimensions they support:

**CSS grid**

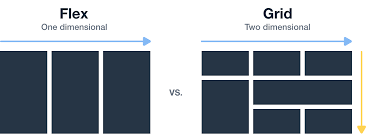
A two-dimensional layout that uses rows and columns to create complex layouts. Grid is ideal for larger layouts that need to be divided into sections.

CSS Grid arranges items in rows and columns (2-Dimension).

**CSS flexbox**

A one-dimensional layout that uses rows or columns to arrange elements in a linear order. Flexbox is more flexible than grid, allowing elements to be moved around on the page.

Flexbox aligns items in a single row or column (1-Dimension).



**Difference Between CSS Grid and Flexbox:**

| **Property** | **Grid** | **Flexbox** |
| --- | --- | --- |
| Dimension | Two – Dimensional | One – Dimensional |
| Features | Can flex combination of items through space-occupying Features | Can push content element to extreme alignment |
| Support Type | Layout First | Content First |
| Primary Use Case | Creating complex layouts with rows and columns | Aligning items in a row or column |
| Performance | Can be less in performance due to very complex grids | Generally faster for simple layouts |

auto-fill vs auto-fit:

In CSS, auto-fill and auto-fit are keywords used to size columns in a grid:

**Auto-fill**

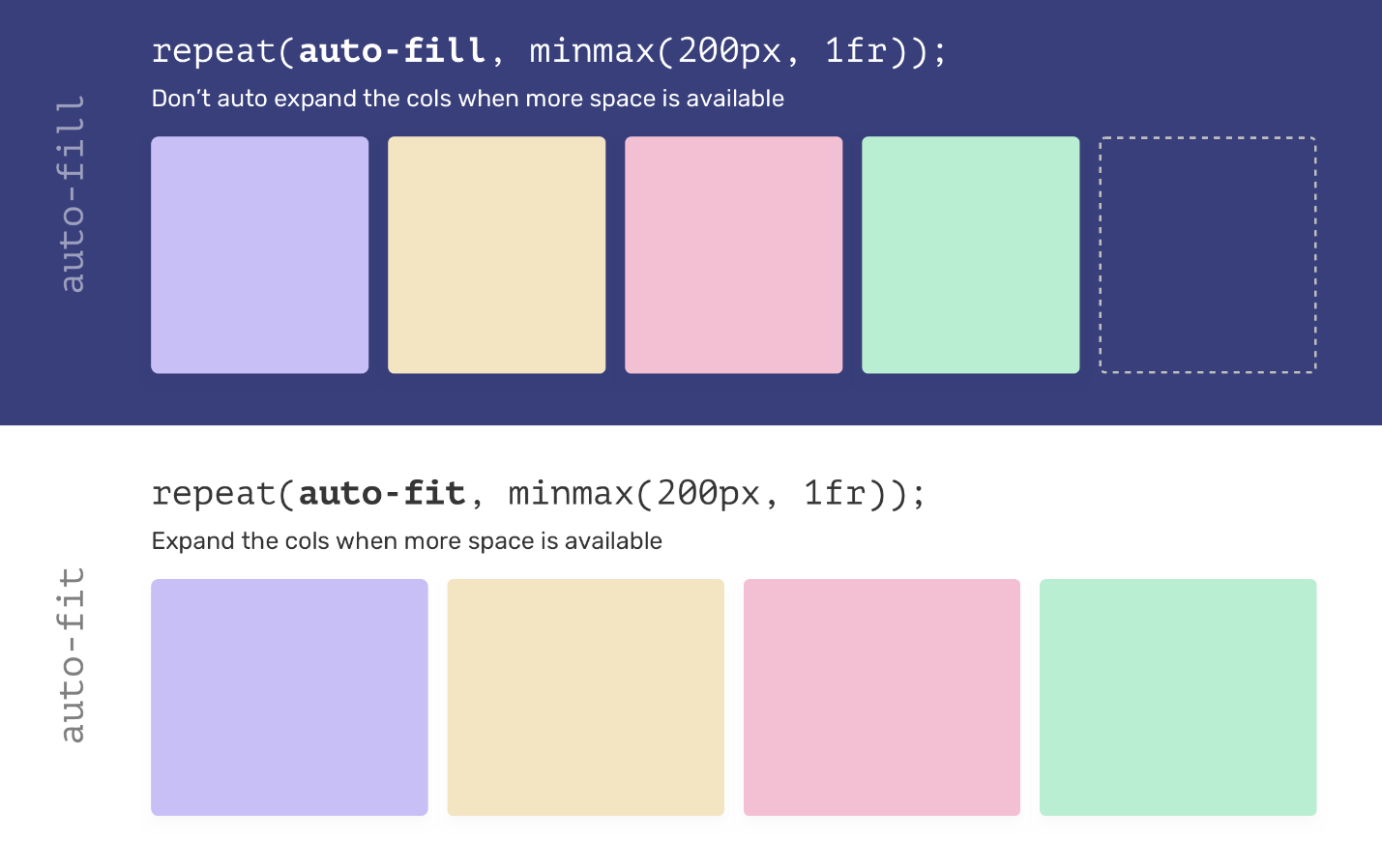
Fills the available space with columns, which may be empty. It does this by adding or removing columns as needed.

The auto-fill property fills the grid with as many columns as it can fit within the available space. If there are more columns than items, the extra columns will be empty but will still occupy space in the grid. This can be useful for maintaining a consistent layout even when the number of items changes.

**Auto-fit**

Fits the current number of columns to the width of the grid. It does this by expanding the columns to take up any available space.

The auto-fit property behaves similarly to auto-fill but with one key difference: it collapses empty columns. This means that if there are fewer items than columns, the empty columns will not take up space, and the remaining columns will expand to fill the available space. This can be useful for creating layouts that adjust dynamically to the number of items.



grid-auto-rows:

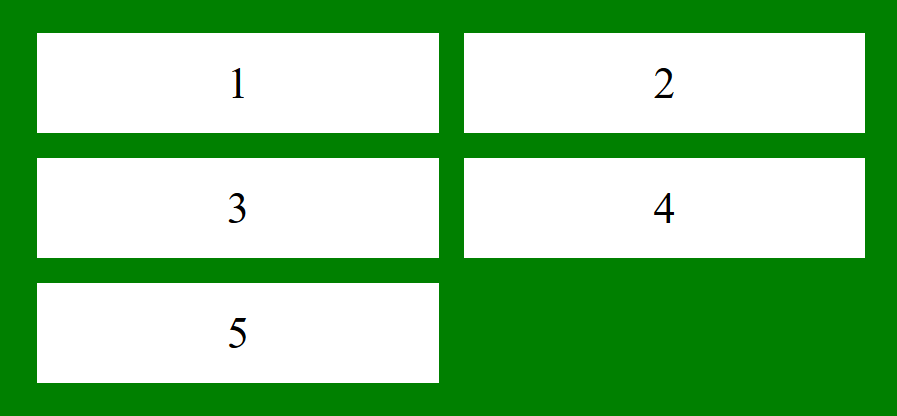
The grid-auto-rows property in Cascading Style Sheets (CSS) specifies the size of rows in a grid container that are created implicitly:

**Definition**: Sets the size of rows that are added to the grid automatically when more rows are needed to contain grid items

**Usage**: Targets rows that were not defined with grid-template-rows or grid-template-areas.

**Syntax:**grid-auto-rows: auto|max-content|min-content|*length*;

The grid-auto-rows CSS property is part of the CSS Grid Layout specification, specifying the size of the grid rows that were created without having an explicit size. In other words, this property sets the size of implicit rows and any other rows that have not been explicitly sized in the grid-template-rows property.



justify-items vs justify-content:

In Cascading Style Sheets (CSS), the properties justify-items and justify-content are used to align items in different ways:

**justify-items**

Aligns grid items along their inline axis. For example, in a 2 x 2 grid layout, the default value for justify-items is stretch, which makes the grid items stretch across the width of their cells. If you hover over the grid container, you can change the value to center, which makes the grid items align in the center of their cells.

**the different values of the justify-items property.**

justify-items: legacy;

justify-items: left;

justify-items: stretch;

justify-items: right;

justify-items: normal;

justify-items: center;

justify-items: start;

justify-items: end;

justify-items: center;

justify-items: flex-end;

justify-items: flex-start;

justify-items: last baseline

**justify-content**

Aligns the items in a flexible container when they don't use all the available space on the main axis (horizontally). For example, you can use justify-content to align a grid along the row axis when the grid is smaller than the space for the grid container. Possible values for justify-content include:

**the different values of the justify-content property.**

justify-content: flex-start;

justify-content: flex-end;

justify-content: center;

justify-content: space-between;

justify-content: space-around;

justify-content: space-evenly;

align-items vs align-content:

In Cascading Style Sheets (CSS), the main difference between align-items and align-content is that align-items controls the alignment of items on a single line, while align-content controls the alignment across multiple lines:

**align-items**

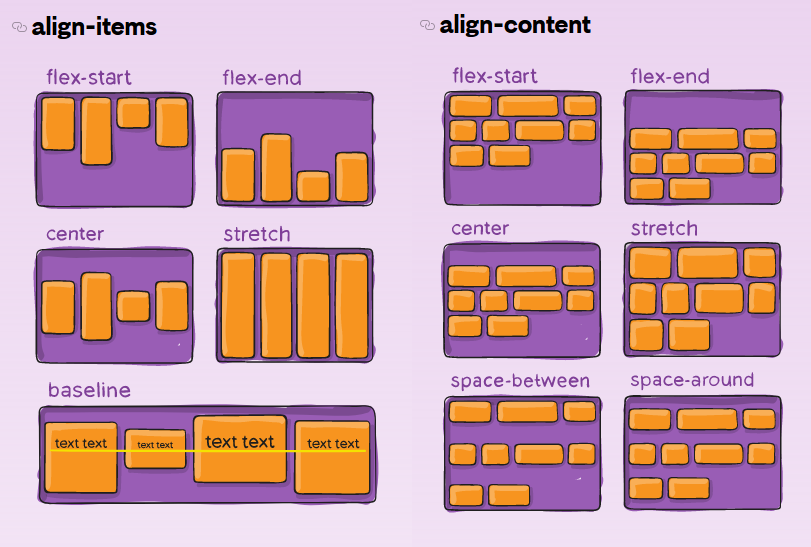
Controls the alignment of items on the cross axis, which is perpendicular to the main axis. align-items positions items within their own row or column.

align-items : Controls the alignment of all items on the cross axis. align-self : Controls the alignment of an individual flex item on the cross axis.

**align-content**

Controls the space between and around flex items on the cross axis. align-content is only used when items wrap into multiple lines of the flex container.

 align-content : Controls the space between flex lines on the cross axis. gap , column-gap , and row-gap : Used to create gaps or gutters between flex items.



grid-auto-flow:

The grid-auto-flow CSS property controls how the auto-placement algorithm works, specifying exactly how auto-placed items get flowed into the grid.

the different values of the grid-auto-flow property.

grid-auto-flow: row;

grid-auto-flow: column;

grid-auto-flow: dense;

grid-auto-flow: row dense;

grid-auto-flow: column dense;

