# slide 2 - Introduction

CSS Grid Layout, is a two-dimensional grid-based layout system that aims to do nothing less than completely change the way we design grid-based user interfaces

CSS has always been used to lay out our web pages, but it is never done a very good job of it. First, we used tables, then floats, positioning and inline-block, but all of these methods were essentially hacks and left out a lot of important functionality.

Flexbox helped out, but it is intended for simpler one-dimensional layouts, not complex two-dimensional ones.

Grid is the very first CSS module created specifically to solve the layout problems

# slide 4- Properties for the Grid Container

## slide 4.1 Display

DISPLAY property defines the element as a grid container and establishes a new grid formatting context for its contents.

* **Value grid** – generates a block-level grid
* **Value inline-grid** – generates an inline-level grid

## slide 4.2 grid-template-columns grid-template-rows

grid-template-columns and grid-template-rows defines the columns and rows of the grid with a space-separated list of values.

## slide 4.3 grid-template-columns grid-template-rows

The fr unit allows you to set the size of a track as a fraction of the free space of the grid container. For example, this will set each item to one third the width of the grid container.

## slide 4.3 grid-template-areas

grid-template-areas defines a grid template by referencing the names of the grid areas which are specified with the [grid-area](https://css-tricks.com/snippets/css/complete-guide-grid/#prop-grid-area) property

Repeating the name of a grid area causes the content to span those cells.

A period signifies an empty cell.

The syntax itself provides a visualization of the structure of the grid.

That will create a grid that is four columns wide by three rows tall.

The entire top row will be composed of the **header** area.

The middle row will be composed of two **main** areas, one empty cell, and one **sidebar** area.

The last row is all **footer**.

## Slide 4.4 - column-gap, row-gap, gap

column-gap, row-gap specifies the size of the grid lines. You can think of it like setting the width of the gutters between the columns and rows. The gutters are only created between the columns and rows, not on the outer edges.

## Slide 4.5 - justify-items

justify-items aligns grid items along the row axis

* **Value start** – aligns items to be flush with the start edge of their cell
* **Value** **end** – aligns items to be flush with the end edge of their cell
* **Value** **center** – aligns items in the center of their cell
* **Value** **stretch** – fills the whole width of the cell (this is the default)

## Slide 4.6 - align-items

align-items aligns grid items along the column axis. Similar to the justify-items

## Slide 4.7 - justify-content

Sometimes the total size of your grid might be less than the size of its grid container.

This could happen if all of your grid items are sized with non-flexible units like px.

In this case you can set the alignment of the grid within the grid container.

This property aligns the grid along the row axis

* **start** – aligns the grid to be flush with the start edge of the grid container
* **end** – aligns the grid to be flush with the end edge of the grid container
* **center** – aligns the grid in the center of the grid container
* **stretch** – resizes the grid items to allow the grid to fill the full width of the grid container
* **space-around** – places an even amount of space between each grid item, with half-sized spaces on the far ends
* **space-between** – places an even amount of space between each grid item, with no space at the far ends
* **space-evenly** – places an even amount of space between each grid item, including the far ends

## Slide 4.7 - align-content

align-content property aligns the grid along the *column* axis with the same way as justify-content.

# slide Properties for the Grid Items

## slide - grid-column-start, grid-column-end, grid-row-start, grid-row-end

grid-column-start, grid-column-end, grid-row-start, grid-row-end determines a grid item’s location within the grid by referring to specific grid lines.

grid-column-start/grid-row-start is the line where the item begins,

and grid-column-end/grid-row-end is the line where the item ends.

## Slide - justify-self

justify-self - aligns a grid item inside a cell along the row axis. This value applies to a grid item inside a single cell.

## Slide - align-self

align-self - aligns a grid item inside a cell along the column axis. This value applies to the content inside a single grid item.

## Slide - MULTIPLE ROWS AND COLUMNS

 Grid can span across multiple rows and columns. Flex by default cannot.

 Grid has templates, gaps and lines. Flex has lines and non-default wrap.

 At the same time, flex can create grid-like layouts with explicit item width and wrap.

## Slide – MIX

 With grid you can avoid media queries for creating responsive layouts if you learn how to use fr units together with min-max function

 You can use grid with flex items. Or you can still use flex-only layouts — items can also be flex containers

 The conclusion is - my be reasonable to use grid for mapping out responsive layout for largest layout areas.

This was overview of CSS Grid, thanks