# 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’s 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 (vertical centering, for instance).

Flexbox helped out, but it’s intended for simpler one-dimensional layouts, not complex two-dimensional ones (Flexbox and Grid actually work very well together)

 Grid is the very first CSS module created specifically to solve the layout problems we’ve all been hacking our way around for as long as we’ve been making websites

# slide 3 - Important Terminology

Before diving into the concepts of Grid it’s important to understand the terminology. Since the terms involved here are all conceptually similar, it’s easy to confuse them with one another if you don’t first memorize their meanings defined by the Grid specification. But don’t worry, there aren’t many of them.

## slide 3.1 Grid Container

The element on which display: grid is applied. It’s the direct parent of all the grid items. In this example container is the grid container.

## slide 3.2 Grid Item

The children (i.e. direct descendants) of the grid container. Here the item elements are grid items, but sub-item isn’t

## slide 3.3 Grid Line

The dividing lines that make up the structure of the grid. They can be either vertical (“column grid lines”) or horizontal (“row grid lines”) and reside on either side of a row or column. Here the yellow line is an example of a column grid line.

## slide 3.4 Grid Cell

The space between two adjacent row and two adjacent column grid lines. It’s a single “unit” of the grid. Here’s the grid cell between row grid lines 1 and 2, and column grid lines 2 and 3.

## slide 3.5 Grid Track

The space between two adjacent grid lines. You can think of them like the columns or rows of the grid. Here’s the grid track between the second and third row grid lines.

## Slide 3.6 Grid Area

The total space surrounded by four grid lines. A grid area may be composed of any number of grid cells. Here’s the grid area between row grid lines 1 and 3, and column grid lines 1 and 3.

# slide 4- Properties for the Parent (Grid Container)

## slide 4.1 Display

Defines the element as a grid container and establishes a new grid formatting context for its contents. Values:

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

## slide 4.2 grid-template-columns 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 4.4 grid-template-columns grid-template-rows

When sizing rows and columns, you can use all the [lengths](https://css-tricks.com/the-lengths-of-css/) you are used to, like px, rem, %, etc, but you also have keywords like min-content, max-content, auto, and perhaps the most useful, fractional units. grid-template-columns: 200px 1fr 2fr min-content;

## slide 4.5 grid-template-columns grid-template-rows

You also have access to a function which can help set boundaries for otherwise flexible units. For example to set a column to be 1fr, but shrink no further than 200px: grid-template-columns: 1fr minmax(200px, 1fr);

## slide 4.6 grid-template-columns grid-template-rows

There is repeat() function, which saves some typing, like making 10 columns: grid-template-columns: repeat(10, 1fr);

## slide 4.3 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’ll create a grid that’s 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

Specifies the size of the grid lines. You can think of it like setting the width of the gutters between the columns/rows. The gutters are only created between the columns/rows, not on the outer edges.

Gap - A shorthand for [row-gap](https://css-tricks.com/snippets/css/complete-guide-grid/#prop-grid-column-row-gap) and [column-gap](https://css-tricks.com/snippets/css/complete-guide-grid/#prop-grid-column-row-gap)

## Slide 4.5 - justify-items

Aligns grid items along the inline (row) axis

## Slide 4.6 - align-items

Aligns grid items along the block (column) axis. This value applies to all grid items inside the container.

This behavior can also be set on individual grid items via the [align-self](https://css-tricks.com/snippets/css/complete-guide-grid/#prop-align-self) property.

## 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 *inline (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

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 *block (column)* 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 height 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