**Welcome to the world of CSS-grid – great tool for interactive web interfaces.**

Let's look at the next two sites and their responsive versions.

In the first case, in design it is easy to see grid from rows and columns.

In the other, grid clearly not visible in design.

But if we look at this websites code, in both cases we will see a lot of squares in one container with property display grid that form a grid.

This is css grid - a new method of placing content on a web page,

This method appeared relatively recently under the name CSS Grid Layout.

This method can be called the return to good old HTML tables layouts, but with modern properties of new generation.

The arrangement webpage’s content like in table is a simple understandable concept for every person. It's like putting things on the shelves in the closet.

But tables layouts contain redundant code, not semantic, weigh a lot and inflexible, allows to create only a strict design.

The order of items in the HTML markup is does not matter.

Grid Layout lets us properly separate the order of elements in the source, in markup, from their visual presentation.

We define the markup of the entire page, and then completely rearrange it to place elements under a different screen width using just a few lines of css-code.

**What about the differences css grid with flexbox?**

Flexbox’s conception like grid, but we have control under cell’s locations only in one dimension – only along row or only along column.

Grid is more flexible tool that operates with cells in two dimensions - along rows and along columns at the same time.

In practice, it is advisable to use, to combine both methods - Grid and Flexbox in one layout

**Browser support.**

Almost all major desktop and mobile browsers - Chrome, Firefox, Opera, MS Edge - can boast CSS support for Grid Layout.

We can easily use CSS grid in IE right now with some nuances.

For example: we ned use instead one property grid-area three another to define grid-area in layout.

**What is a grid**

The general algorithm of CSS Grid is simple.

A grid is an intersecting set of horizontal and vertical lines – one set defining columns, and the other, rows.

If an element of a web page has a property display: grid, then the CSS Grid browser module automatically draws a table according to the explicitly or by default specified properties.

When we start working with css grid, we create container like parent element.

It’s the Grid container, for which we declare property display: grid or display: inline-grid on an element (determines container’s location with its neighboring elements)

All direct children of that element become grid items.

By default, if all the grid parameters are not explicitly set, the CSS Grid module will draw a table in one column. By default, there are no spaces between the cells.

In this example, I have a containing div with a class “wrapper” and, inside are 12 child elements.

**Grid tracks**

Rows and columns in grid define grid tracks - horizontal or vertical.

To set the size and count of columns and rows it’s to need using properties grid-template-columns and grid-template-rows.

We can explicitly set a count and size of columns and rows like in 1 example.

Or we can write it more short, laconic with notation repeat.

In huge grid-tables with a large number of cells, the notation repeat() will often help out.

But this is not the limit of compactness. Like many cascading style sheet definitions,

css grid properties have abbreviated notation.

Columns and lines can be described by one property grid-template.

What if we do not know in advance exactly how many rows of elements will be on our page.

If there are more elements in the container than explicitly specified in the grid definition,

then the CSS Grid module will simply self-draw new cells.

Automation can be controlled through the properties grid-auto-rows, grid-auto-columns.

We can create a grid with fixed or flexible track sizes – using pixels for example.

Unit fr designed for this purpose.

It represents a fraction of the available space in the grid container.

fr is calculated based on the remaining space when combined with other length values.

Also Tracks sizes can be defined to have a minimum and/or maximum size with the minmax() function.

And combine all variations.

**Spaces between rows and columns** can be created using the [grid-column-gap](https://developer.mozilla.org/en-US/docs/Web/CSS/grid-column-gap), [grid-row-gap](https://developer.mozilla.org/en-US/docs/Web/CSS/grid-row-gap), or [grid-gap](https://developer.mozilla.org/en-US/docs/Web/CSS/grid-gap) properties.

grid-gap is a shorthand for grid-row-gap and grid-column-gap, and it’s very straightforward to use.

The grid-gap properties are not the only thing that can cause tracks to space out.

Margins, padding or the use of the space distribution properties in Box Alignment can all contribute to the visible gap.

Therefore the grid-gap properties should not be seen as equal to “the gutter size”.

We can change them in any point of layout with one of css methods

**Grid lines and cells**

are the lines that make up the grid.

The rows in the CSS Grid tables are bounded by horizontal lines, and the columns are vertical.

Each line in grid has its own number by which it can be accessed.

the space between 4 grid lines – is Grid Cell like classic table’s cell.

The highlighted grid cell in this image is between row lines 2 and 3 and column lines 2 and 3.

So it is the smallest unit.

**Grid area - merging several cells.**

Like in HTML tables, it was possible to merge cells through the colspan and rowspan attributes. in the CSS Grid merging cells is also possible using names of lines. The merged cells are called grid area.

There is also a short version of this entry through the grid-area property.

**Z-index**

Unlike HTML tables in CSS Grid, cells and areas are located not only in the plane - they can also be located in a “stack”, like layers in Photoshop.

Each layer can be assigned a number through the special z-index property.

This allows you to use the z-index property for the effect of overlapping regions and cells on each other.

**And about alignment control**

CSS’s [Box Alignment Module](https://drafts.csswg.org/css-align/) complements CSS Grid to allow items to be aligned along the row of column axis.

justify-items and justify-self align items along the row axis, and align-items and align-self align items along the column axis.

justify-items and align-items are applied to the grid container and support the following values

Individual items can be self-aligned with the align-self and justify-selfproperties.

These properties support the following values.