**Вопросы 💎**

1. Что за единица измерения - fr?

Grid introduces an additional length unit to help us create flexible grid tracks. The new fr unit represents a fraction of the available space in the grid container. CSS Grid includes a method that can help save time calculating percentages, this method involves setting track widths using units of measure fr. This is a flexible unit of measure that is calculated based on the amount of available space in the grid container.

The next grid definition would create three equal width tracks that grow and shrink according to the available space:

.wrapper {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

}

In this next example, we create a definition with a 2fr track then two 1fr tracks. The available space is split into four. Two parts are given to the first track and one part each to the next two tracks.

.wrapper {

display: grid;

grid-template-columns: 2fr 1fr 1fr;

}

In this final example, we mix absolute sized tracks with fraction units. The first track is 500 pixels, so the fixed width is taken away from the available space. The remaining space is divided into three and assigned in proportion to the two flexible tracks.

.wrapper {

display: grid;

grid-template-columns: 500px 1fr 2fr;

}

1. Как можно задать грид с 5 колонками шириной по 20%? Минимум 2 способа.

**1 way:**

**<div class="container">**

<div class="item item1"></div>

<div class="item item2"></div>

<div class="item item3"></div>

<div class="item item4"></div>

<div class="item item5"></div>

**</div>**

**.container {**

display: grid;

grid-template-columns: 20% 20% 20% 20% 20%;

**}**

**Or alternatively instead of repeating 20% several times:**

grid-template-columns: repeat(5, 20%);

**2 way:**

.container {

            display: grid;

            width: 100%;

            grid-template-areas: "column1 column2 column3 column4 column5"

        }

<div class="container">

        <div class="column">Content1</div>

        <div class="column">Content2</div>

        <div class="column">Content3</div>

        <div class="column">Content4</div>

        <div class="column">Content5</div>

    </div>

**3 way:**

.container {

            display: grid;

            width: 100%;

            grid-template-columns: 1fr 1fr 1fr 1fr 1fr;

        }

<div class="container">

        <div class="column">Content1</div>

        <div class="column">Content2</div>

        <div class="column">Content3</div>

        <div class="column">Content4</div>

        <div class="column">Content5</div>

    </div>

**4 way:**

.container {

            display: grid;

            width: 100%;

            grid-template-columns: repeat(auto-fit, minmax(20%, 1fr));

        }

<div class="container">

        <div class="column">Content1</div>

        <div class="column">Content2</div>

        <div class="column">Content3</div>

        <div class="column">Content4</div>

        <div class="column">Content5</div>

    </div>

1. Самостоятельно разберитесь, что такое auto-fill и auto-fit ?

To achieve wrapping, we can use the **auto-fit** or **auto-fill** keywords. The code looks like this:

grid-template-columns: repeat( auto-fit, minmax(250px, 1fr) );

auto-fill FILLS the row with as many columns as it can fit. So it creates implicit columns whenever a new column can fit, because it’s trying to FILL the row with as many columns as it can. The newly added columns can and may be empty, but they will still occupy a designated space in the row.

auto-fit FITS the CURRENTLY AVAILABLE columns into the space by expanding them so that they take up any available space. The browser does that after FILLING that extra space with extra columns (as with auto-fill ) and then collapsing the empty ones.

1. Как сделать такую табличку? Параметры: первая колонка шириной 100 пикселей, вторая 30%. Первая строчка высотой 200 пикселей, вторая строчка 100 пикселей.
2. .container {
3. display: grid;
4. grid-template-columns: 100px 30% 1fr;
5. grid-template-rows: 200px 100px 1fr;
6. gap: 15px;
7. }
8. <div class="container">
9. <div class="item item1">First cell</div>
10. <div class="item item2">Second cell</div>
11. <div class="item item3">Third cell</div>
12. <div class="item item4">Fourth cell</div>
13. <div class="item item3">Fifth cell</div>
14. <div class="item item4">Sixth cell</div>
15. </div>
16. Как сделать такое выравнивание в грид-контейнере?

It is important to use the „space-between“ value for the justify-content property.

.container {

display: grid;

width: 100%;

height: 500px;

grid-template-columns: 2fr 1fr 2fr;

grid-template-rows: 1fr 2fr 1fr;

justify-content: space-between;

grid-column-gap: 20%;

}

1. Что такое и как задается *grid area*?

Grid area is a space within a grid container in which one or more grid elements can be placed. This element can consist of one or more grid cells.

The `grid-area` property is a shorthand property for `grid-row-start`, `grid-column-start`, `grid-row-end` and `grid-column-end`, defining the size and location of the grid element.

.box1 { grid-area: 2 / 1 / 4 / 3; }

1. Приведите пример использования grid-template-areas (не копированием из этого урока 😉)

Another way to define a grid area involves placing our items using named template areas, and define its location as the values of the grid-template-areas property. You can select any name for the grid area.

The default grid layout, on the other hand, places the grid elements in rows based on a start and end point (grid-column-start and grid-column-end).

Each area in inverted commas defines a row and each word in that row defines a column.

If a name is repeated in adjacent cells (whether next to or below each other), the element extends across both or more cells.

To make an area cover more than one cell when describing the grid structure with the grid-template-areas property, you must rewrite its identifier several times along the row or column. The created area must be a rectangle, L and T-shaped areas are not allowed. It is also possible to create only one rectangular area per name - disconnected areas are not allowed.

Example:

.grid-raster {

display:grid;

gap: 14px;

grid-template-columns: 1fr 1fr 1fr;

grid-template-areas: "radis plum mais"

"erbse pilz mais";

column-gap: 14px;

row-gap: 24px;

}

Grid-area moves the elements into the appropriate cells.

.radis {

grid-area: radis

}

.plum {

grid-area: plum

}

.mais {

grid-area: mais

}

.erbse {

grid-area: erbse

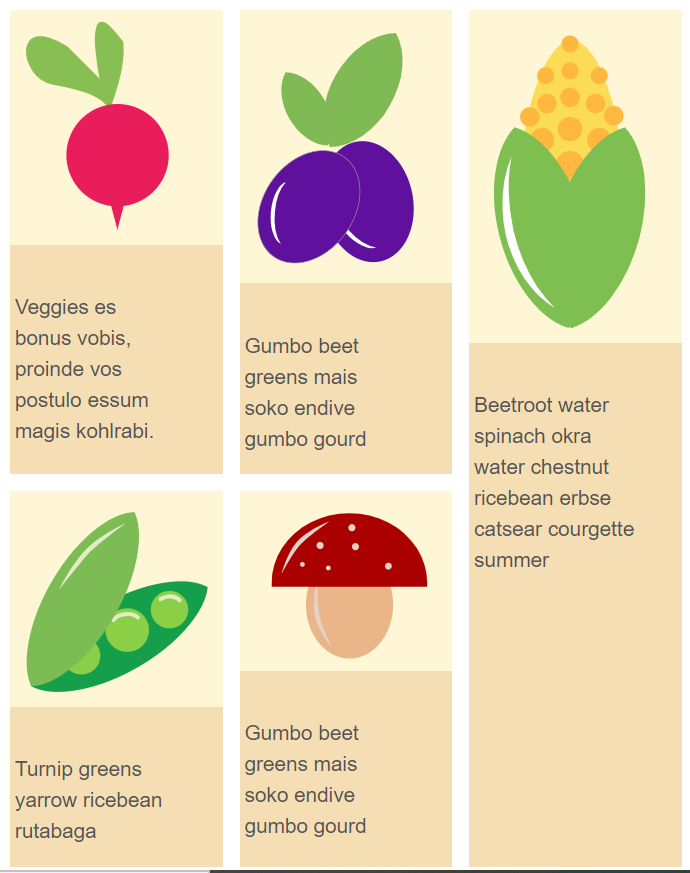
}

.pilz {

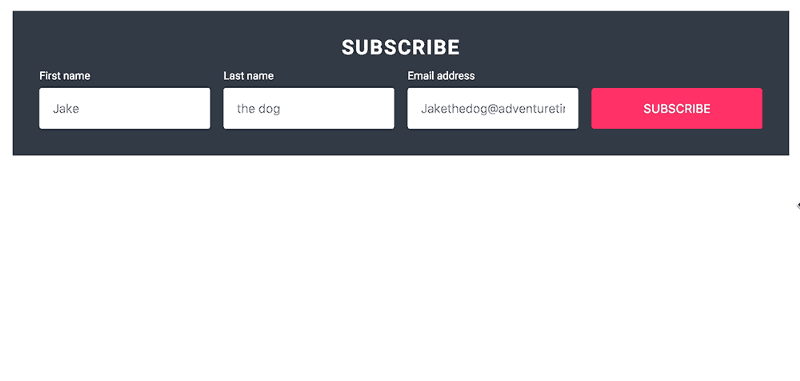
grid-area: pilz

}

The areas are marked by meaningful names, a dot stands for an empty cell.



1. Каким свойством можно задать такое поведение элементов?



Ссылка на анимацию: <http://css-live.ru/Primer/grid1/1-MyO6enrNZEL9HSzYmdA8VA.gif>

We can apply here auto-fit, in this case after placing the grid items any empty repeated tracks are collapsed. An empty track is one with no in-flow grid items placed into or spanning across it. (This can result in all tracks being collapsed, if they’re all empty.)

A collapsed track is treated as having a single fixed track sizing function of 0px, and the gutters on either side of it collapse.

1. Самостоятельно разберитесь, как работают именованные линии? Есть ли какие-то рекомендованные правила наименований? Если да, то какие?

Let's start by naming the grid lines. In the example below there is a Grid with six explicit column tracks and one explicit row track. Elements are placed on the grid using line numbers.

.grid {

display: grid;

grid-gap: 20px;

grid-template-rows: 20vh ;

grid-template-columns: 1fr 2fr 1fr 2fr 1fr 2fr;

}

.header {

grid-row: 1;

grid-column: 1 / -1;

}

.sidebar {

grid-row: 2;

grid-column: 1 / 3;

}

.content {

grid-row: 2;

grid-column: 3 / -1;

}

If necessary, line names can be given inside square brackets in the track list. Here it is important to remember that you are naming the line, not the track that follows it. Having named the lines, you can use their names instead of numbers when positioning items on the grid.

.grid {

display: grid;

grid-gap: 20px;

grid-template-rows: [header-start] 20vh [header-end] ;

grid-template-columns: [sidebar-start] 1fr 2fr [sidebar-end] 1fr 2fr 1fr 2fr;

}

.header {

grid-row: header-start;

grid-column: 1 / -1;

}

.sidebar {

grid-row: 2;

grid-column: sidebar-start / sidebar-end;

}

A good practice is to name using the suffix **-start** for the start lines (whether they are line or column lines) and **-end** for the end lines. The result should be main-start and main-end or sidebar-start and sidebar-end.

Often the end line of one part of the grid and the start line of another part overlap, but this is not a problem, as lines can have multiple names. You can give a line more than one name by adding a space inside the square brackets.

.grid {

display: grid;

grid-gap: 20px;

grid-template-rows: [header-start] 20vh [header-end] ;

grid-template-columns: [full-start sidebar-start] 1fr 2fr [sidebar-end main-start] 1fr 2fr 1fr 2fr [main-end full-end];

}

.header {

grid-row: header-start;

grid-column: full-start / full-end;

}

.sidebar {

grid-row: 2;

grid-column: sidebar-start / sidebar-end;

}

.content {

grid-row: 2;

grid-column: main-start / main-end;

}

There are also situations where several lines have the same name. This happens if you use the repeat() function and include named lines in the track list. The following example creates six named lines, alternately named col-a-start and col-b-start.

.grid {

display: grid;

grid-gap: 20px;

grid-template-columns: repeat(3, [col-a-start] 1fr [col-b-start] 2fr);

}

If you specify col-a-start when positioning an element, it will be placed on the first line of the grid you encounter named col-a-start (which would be the first grid line in the example above). By the same token, if you specify col-b-start when positioning an element, it will be placed on the second grid line.

If you want to place the item on subsequent lines, add a number after the name to indicate on which line with this name the item should be placed.

.box3 {

grid-row: 2;

grid-column: col-a-start 2 / col-b-start 3;

}

1. Как проще всего задать 12 одинаковых по ширине колонок?

When using the repeat() function, the value before the comma specifies the number of repetitions for the list of tracks following the comma. This track list can contain many values, so you can set a particular set of tracks to repeat in this way.

In our case, the code will lool like this:

grid-template-columns: repeat(12, 1fr);