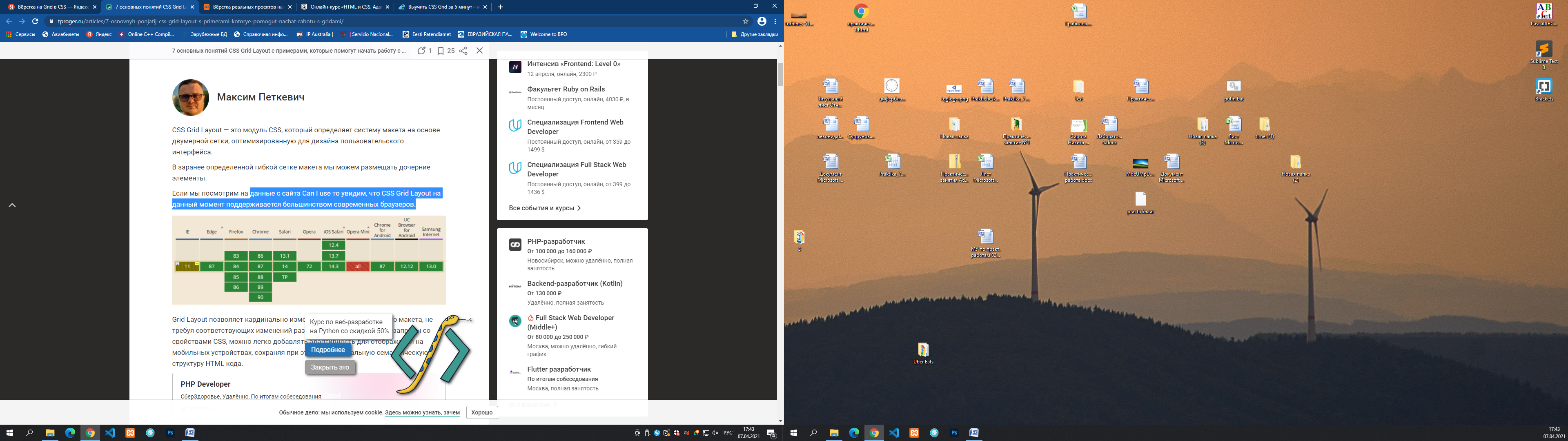
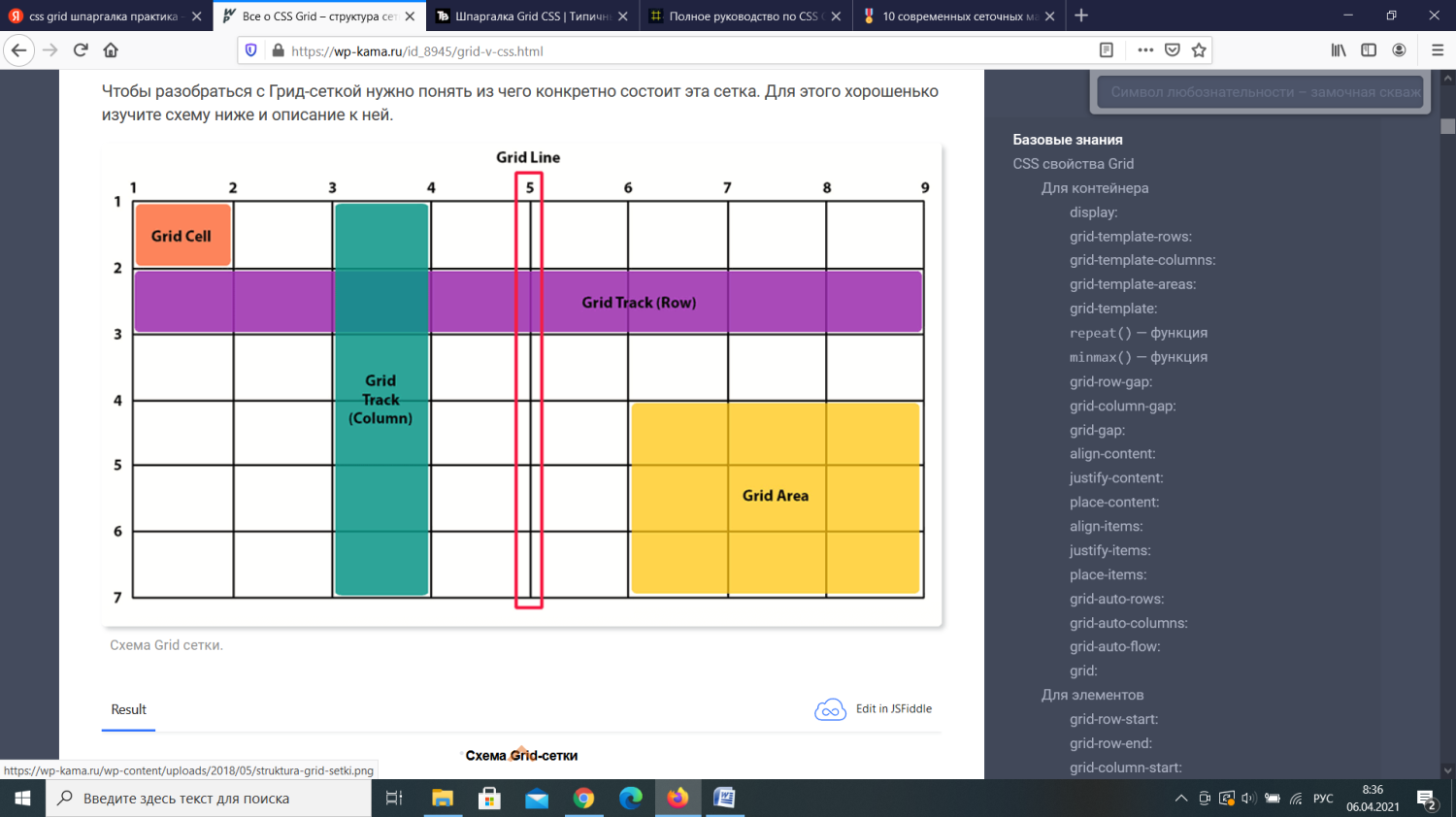
Теория к практическому занятию

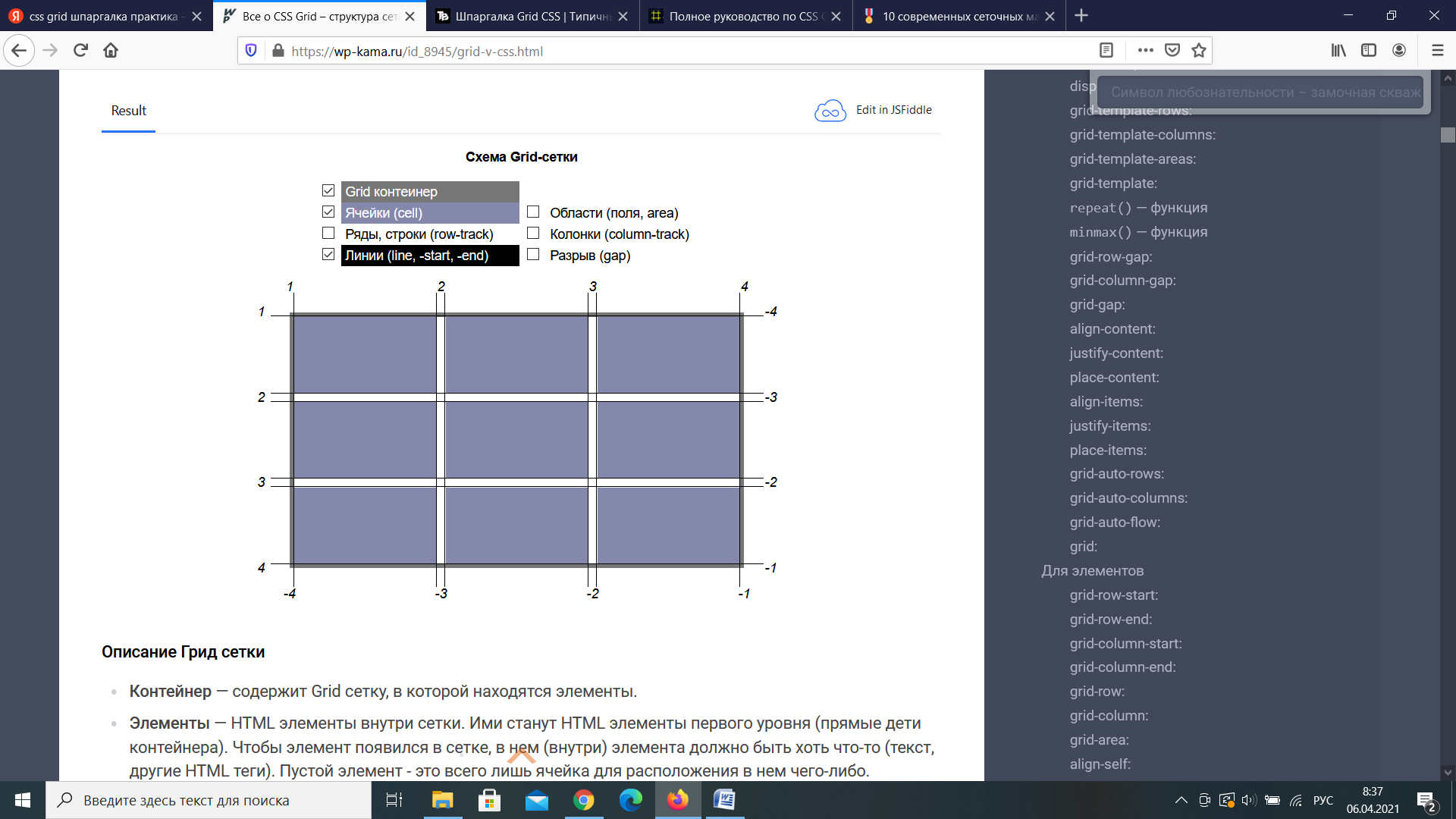
**Вёрстка на Grid в CSS**

## Grid — это набор горизонтальных и вертикальных «линий», которые пересекаются между собой и создают сетку из рядов и колонок. Элементы могут быть помещены в сетку, опираясь на номер линии или номер ряда/колонки.

Данные с сайта **Can I use** подтверждают, что CSS Grid Layout на данный момент поддерживается большинством современных браузеров.







##### **Описание Грид сетки**

* **Контейнер** — содержит Grid сетку, в которой находятся элементы.
* **Элементы** — HTML элементы внутри сетки. Ими станут HTML элементы первого уровня (прямые дети контейнера). Чтобы элемент появился в сетке, в нем (внутри) элемента должно быть хоть что-то (текст, другие HTML теги). Пустой элемент - это всего лишь ячейка для расположения в нем чего-либо.
* **Линии** — это образные линии (на самом деле никаких линий нет), разделяющие сетку на колонки и ряды, они создают структуру сетки. Линии автоматически нумеруются. Также линиям можно указывать имена, чтобы потом прикреплять к ним элементы по номеру или по имени линии. Линия — это номер или имя колонки/ряда. Расстояние между линиями (колонками/рядами) можно указать через grid-gap:, grid-row-gap:, grid-column-gap:.
* **Ряд/колонка (row/column, track)** — все что находится между соседними линиями, т.е. линии разделяют сетку на ряды и колонки.
* **Ячейка (cell)** — место куда будет расположен элемент. Ячейка это пересечение колонки и ряда.
* **Область (area, поле)** — объединение одной или нескольких ячеек в общую ячейку (поле). Это такая большая ячейка также ограниченная линиями. Области можно задать имя, чтобы удобно было помещать туда элементы.
* **Разрыв (gap)** — расстояние между рядами и колонками. Разрывает линию на две. Так между линиями, а как следствие и колонками/рядами, ячейками появляется пустое место. Это своего рода margin, border-spacing между ячейками. По умолчанию линия между ячейками всего одна (ячейки слеплены), но если указать разрыв, то мы разорвем линию, и появится расстояние между колонками/рядами, при этом номер или имя линии (колонки/ряда) остается одно.

**Для рассмотрения данной темы запишите образец кода index.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="style.css">

<title>CSS Grid</title>

</head>

<body>

<div class="grid-init grid">

<div class="box-init box">A</div>

<div class="box-init box">B</div>

<div class="box-init box">C</div>

<div class="box-init box">D</div>

<div class="box-init box">E</div>

<div class="box-init box">F</div>

</div>

</body>

</html>

Образец кода **style.html**

\* {

/\* width и height элементов включают в

себя значения полей и границ \*/

box-sizing: border-box;

}

.grid-init {

max-width: 400px; /\* максимальная ширина контейнера \*/

margin: auto; /\* центрирование контейнера на странице \*/

background-color: lightsteelblue; /\* фон \*/

padding: 10px; /\* внутренниеотступы \*/

}

.box-init {

font-size: larger; /\* размершрифта \*/

color: #fff; /\* цветтекста \*/

border-radius: 5px; /\* скруглениеуглов \*/

background-color: #2196f3; /\* цветфона \*/

border: 1px solid black; /\* границаблока \*/

padding: 10px; /\* внутренние отступы \*/

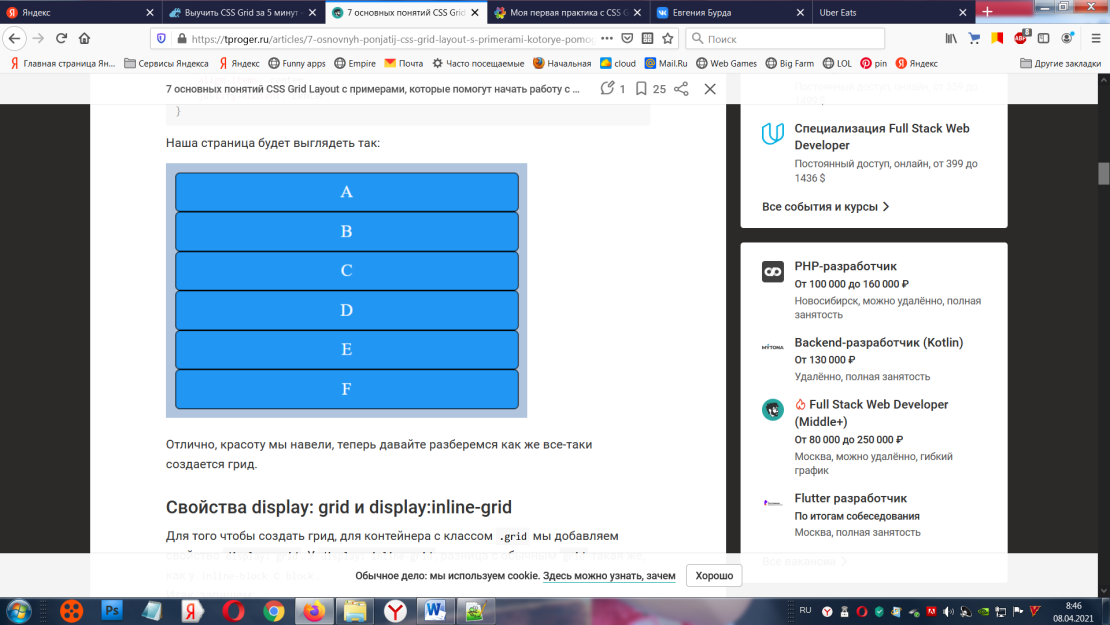
display: flex; /\* выравнивание текста по центру блока с помощью CSS Flexbox \*/

align-items: center;

justify-content: center;

}

Страница будет выглядеть так:



**Свойства display: grid и display:inline-grid**

Для того чтобы создать грид, для контейнера с классом .grid добавим свойство display: grid. У display: inline-grid, разница с обычным grid такая же, как у inline-block c block.  
запишем:

.grid {

display: grid;

}

С этого момента контейнер **.grid** будет являться гридом, визуально разницы пока не увидим.

## Свойства grid-gap, row-gap и column-gap

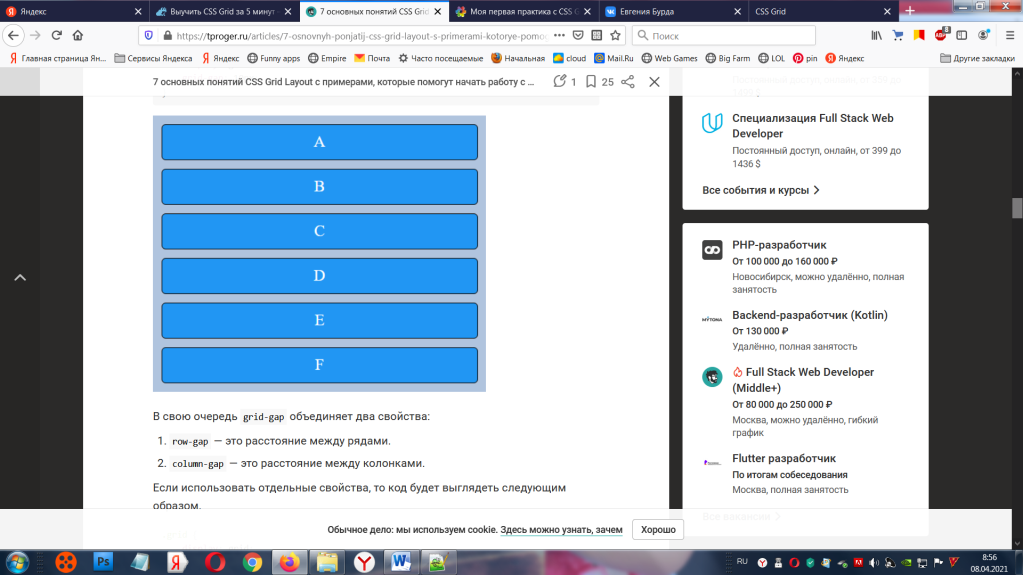
Для задания отступов между ячейками грида используется grid-gap. Задайте ему значение в 10px.

.grid {

display: grid;

grid-gap: 10px;

}



В свою очередь grid-gap объединяет два свойства:

1. row-gap — это расстояние между рядами.
2. column-gap — это расстояние между колонками.

Если использовать отдельные свойства, то код будет выглядеть следующим образом.

.grid {

display: grid;

row-gap: 10px;

column-gap: 10px;

}

## Свойство grid-template-columns

Добавим колонки.

.grid {

display: grid;

grid-gap: 10px;

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

}

Ширину мы можем указывать в любых CSS единицах длины, в процентах и в единицах **fr**.

fr (от fraction — часть) это новая форма единицы измерения созданная для гридов, которая позволяет добавлять столько колонок, сколько мы захотим, не заботясь о конкретном значении ширины колонки. Свободное место распределяется между такими колонками пропорционально количеству fr.

1fr 1fr 1fr — такая запись означает, что мы создаем 3 колонки одинаковой ширины.

***Немного математики:***

Ширина грида равна 400px. Если мы зададим для grid-template-columns следующие значения: 1.5fr 2fr 1fr, то ширина всех колонок будет равна 1.5fr + 2fr + 1fr= 4.5fr.

Теперь вычислим ширину каждой колонки. Первая колонка будет иметь ширину 1.5fr / 4.5fr \* 400px. Что в итоге равно 133px. Соответственно рассчитаем остальные значения.

Вторая колонка 2fr / 4.5fr \* 400px = 178px, третья колонка 1fr / 4.5fr \* 400px = 89px. Т.е. мы могли задать колонки таким образом:

grid-template-columns: 133px 178px 89px;

В сумме общая ширина равна 400px.

## Свойство grid-template-rows

Это свойство позволяет добавлять строки в грид.

.grid {

display: grid;

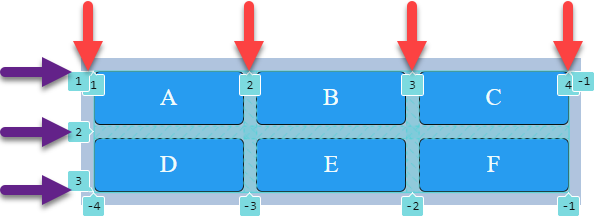
grid-gap: 10px;

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

grid-template-rows: 1fr 1fr;

}

Рассмотрим подробнее, что из себя представляет наш грид и разберем несколько новых понятий.

[](https://tproger.ru/s3/uploads/2021/02/6.png)

***Грид-контейнер*** — элемент, в котором находится сетка грида.

***Грид-линии*** — невидимые вертикальные и горизонтальные линии, разделяющие грид на ячейки. У грид линий есть нумерация, а также им можно задавать имена. На изображении помечены красными (вертикальные линии) и фиолетовыми стрелками (горизонтальные линии).

***Грид-полосы*** — пространство, которое ограничено парой соседнихгрид-линий. Бывают вертикальные и горизонтальные.

***Грид-ячейки*** — то, что получается на пересечении двух грид-полос. По аналогии с ячейками таблицы. На картинке это синие блоки с буквами, в количестве шести штук.

***Грид-области*** — прямоугольники из смежныхгрид-ячеек. Каждая грид-область ограничена двумя парами грид-линий (парой вертикальных и парой горизонтальных).

***Грид-интервалы*** — пустые пространства между соседними грид-полосами.

**Свойства grid-auto-rows, grid-auto-columns и grid-auto-flow**

В примере создания колонок мы не задавали колонки. Это произошло потому, что колонки и строки могут быть заданы в явном и в неявном виде.

Если не указать grid-template-rows, это значит, что в случае надобности строки будут добавляться автоматически и будут неявными.

Высота строки на неявных гридах должна выставляться с помощью свойства grid-auto-rows.

В свою очередь для колонок есть свойство grid-auto-columns которое выставляет ширину колонки.

Свойство grid-auto-flow позволяет указать, что нужно использовать для авто-размещения элементов, строки или колонки. По умолчанию его значение равно row.

Рассмотрим пример, показывающий неявное добавление колонок. Для этого задайте grid-auto-flow равным column. Также уменьшите количество явно заданных колонок до двух. В этом примере в последней ячейке пропишем текст FF.

.grid {

display: grid;

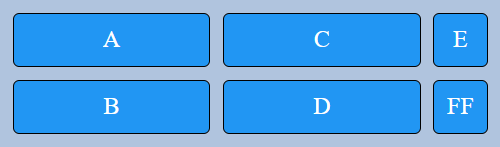
grid-gap: 10px;

grid-template-columns: 1fr 1fr;

grid-template-rows: 1fr 1fr;

grid-auto-flow: column;

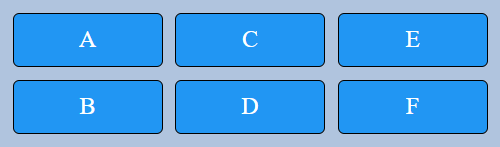
}

[](https://tproger.ru/s3/uploads/2021/02/7.png)

***Важно***, что ячейки выстраиваются вдоль колонок, а не строк. A и B в первой колонке, C и D во второй и т.д.

Видно, что третья колонка добавилась автоматически и ее ширина формируется на основе контента в самой широкой ячейке.Укажитеширину для этой колонки.

grid-auto-columns: 1fr;

[](https://tproger.ru/s3/uploads/2021/02/8-1.png)

Получаем уже привычный грид.

**Функция repeat()**

Эта функция позволяет сокращать объявление колонок и строк в случае повторяющихся значений.

Например есть такой код.

.grid {

display: grid;

grid-gap: 10px;

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

grid-template-rows: 1fr 1fr;

}

Используя функцию repeat мы можем переписать наш код следующим образом.

.grid {

display: grid;

grid-gap: 10px;

grid-template-columns: repeat(3, 1fr);

grid-template-rows: repeat(2, 1fr);

}

**Функция minmax(), auto-fill и auto-fit**

Рассмотрим пример с использованием этой функции.

.grid {

display: grid;

grid-gap: 10px;

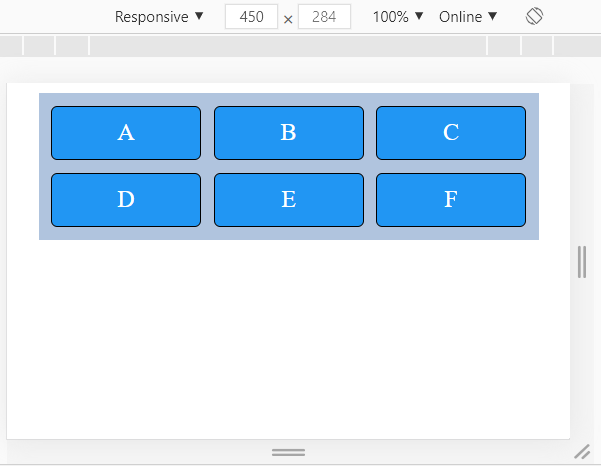
grid-template-columns: repeat(auto-fill, minmax(120px, 1fr));

}

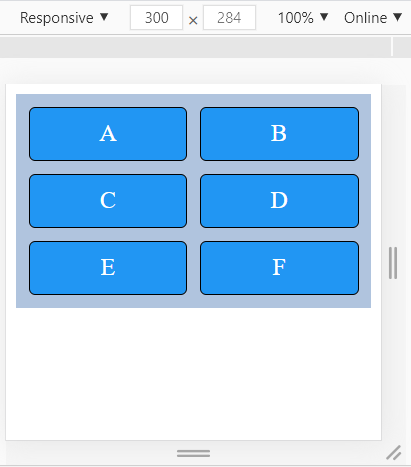
Запись minmax(120px, 1fr) означает, что минимальный размер столбца равен 120px, а максимальный — 1fr.

auto-fill делает наш грид адаптивным, то есть в нем элементы меняют свои позиции с тем, как мы меняем размер окна браузера.

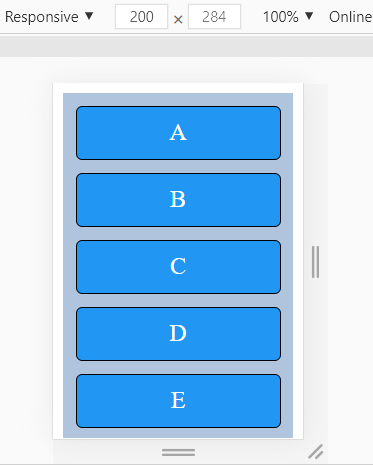
Ширина окна браузера 450px:

[](https://tproger.ru/s3/uploads/2021/02/9.png)

Ширина окна браузера 300px:

[](https://tproger.ru/s3/uploads/2021/02/10.png)

Ширина окна браузера 200px:

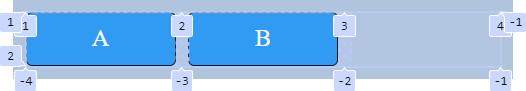
[](https://tproger.ru/s3/uploads/2021/02/11.png)

auto-fit в многострочныхгридах по своему действию похож на auto-fill. Разница проявляется в однострочных гридах. С auto-fit пустые полосы сжимаются до нуля, т.е. они как будто исчезают.

Чтобы показать разницу, рассмотрим следующие примеры:

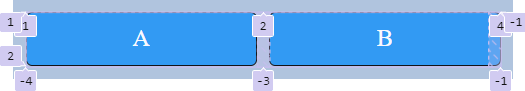
С auto-fill у нас три колонки

grid-template-columns: repeat(auto-fill, minmax(120px, 1fr));

[](https://tproger.ru/s3/uploads/2021/02/12.png)

С auto-fit пустая колонка исчезает

grid-template-columns: repeat(auto-fit, minmax(120px, 1fr));

[](https://tproger.ru/s3/uploads/2021/02/13.png)

**Свойства grid-template-areas, grid-area**

Есть разные способы создания грид-ячеек и грид-областей. Один из способов — это использование грид-линий, за это отвечают свойства grid-column-start, grid-column-end, grid-row-start и grid-row-end. При таком подходе код довольно сложно читать и в случае серьезного изменения макета, придется переписывать большой объем кода. Поэтому рассмотрим второй способ — использование grid-template-areas и grid-area.

Для начала отредактируем html-код и используем в нем семантические теги.

<div class="grid-init grid">

<header class="box-init box l-header">HEADER</header>

<nav class="box-init box l-nav">NAV</nav>

<main class="box-init box l-main">MAIN</main>

<aside class="box-init box l-aside">ASIDE</aside>

<section class="box-init box l-section">SECTION</section>

<footer class="box-init box l-footer">FOOTER</footer>

</div>

C помощью grid-area зададим псевдонимы для наших блоков.

.l-header {

grid-area: my-header;

}

.l-nav {

grid-area: my-nav;

}

.l-main {

grid-area: my-main;

}

.l-aside {

grid-area: my-aside;

}

.l-section {

grid-area: my-section;

}

.l-footer {

grid-area: my-footer;

}

С помощью grid-template-areas определим шаблон страницы.

.grid {

display: grid;

grid-gap: 10px;

grid-template-areas:

"my-headermy-headermy-header"

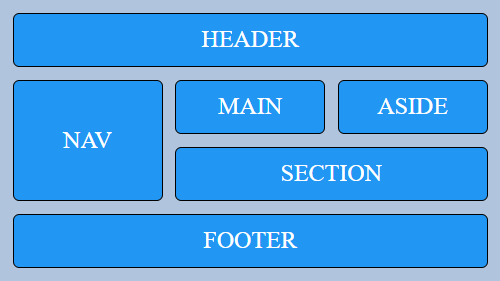
"my-nav my-main my-aside"

"my-nav my-section my-section"

"my-footer my-footer my-footer";

grid-template-columns: repeat(auto-fit, minmax(120px, 1fr));

}

[](https://tproger.ru/s3/uploads/2021/02/14.png)

У нас получился грид 3×4 (3 колонки, 4 ряда). Также у нас есть 6 грид-областей.  
Можноизменить шаблон просто изменив грид-области в grid-template-areas.

.grid {

display: grid;

grid-gap: 10px;

grid-template-areas:

"my-nav my-header my-header"

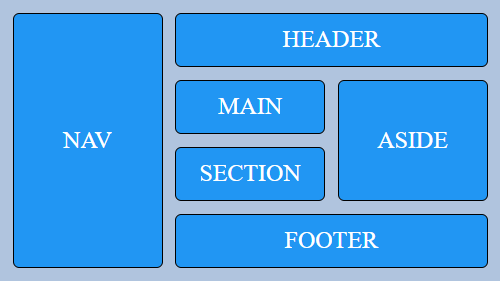
"my-nav my-main my-aside"

"my-nav my-section my-aside"

"my-nav my-footer my-footer";

grid-template-columns: repeat(auto-fit, minmax(120px, 1fr));

}

[](https://tproger.ru/s3/uploads/2021/02/15.png)

Источник:https://tproger.ru/articles/7-osnovnyh-ponjatij-css-grid-layout-s-primerami-kotorye-pomogut-nachat-rabotu-s-gridami/