CSS Grid.

Общая информация. Именование элементов. Свойства grid. Grid-template. Объединение элементов. Сокращенные свойства.

Общая информация

Мы уже знакомы с одним из подходов к адаптивному размещению элементов на странице – Flexible Box. Он позволяет манипулировать элементами на странице, распределяя их вдоль главной и второстепенной осей.

Но, flex-ы не единственный способ распределить элементы по странице. Рассмотрим использование модуля CSS Grid для позиционирования блоков.

Элементы Грида (grid items) располагаются вдоль главной или основной (main) и поперечной (cross) оси. При помощи различных свойств мы можем манипулировать элементами для создания макетов.

Помимо прочего, у нас имеется возможность объединять строки и колонки подобно тому, как мы это делаем в таблицах, что предоставляет нам большую гибкость, чем Flexbox.

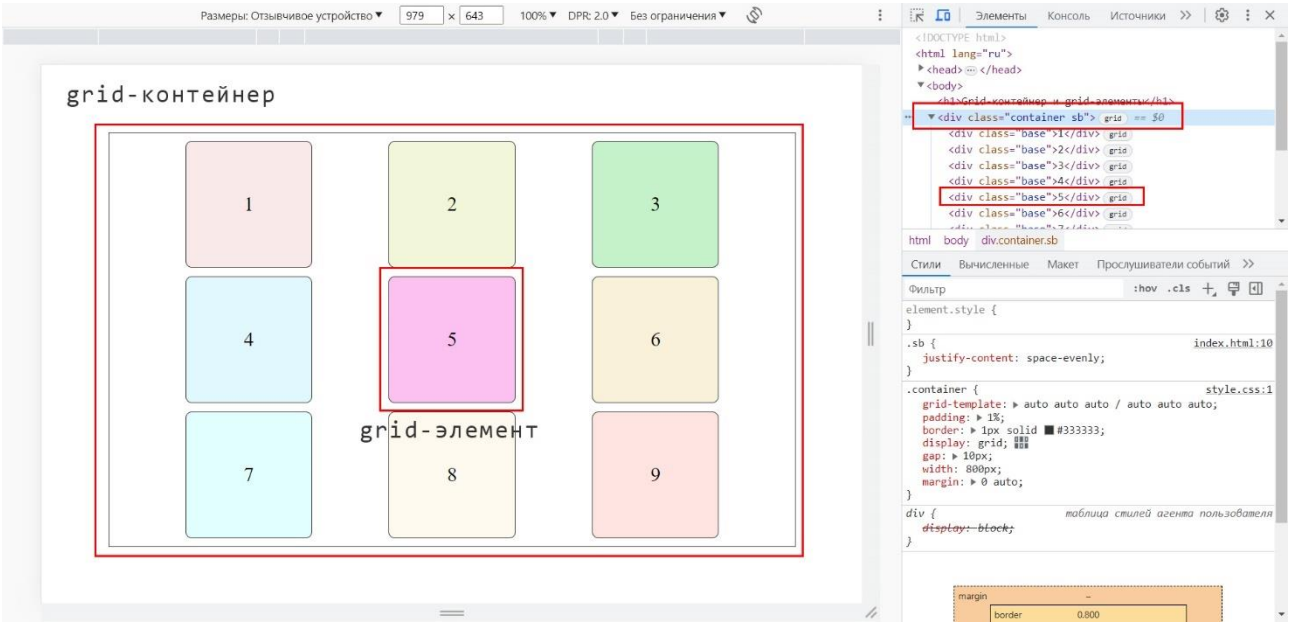
Начнем с терминов, которыми необходимо оперировать при использовании гридов.

Грид — это двумерная сетка. Как табличная, только виртуальная, не связанная с разметкой.

Состоит из следующих элементов:

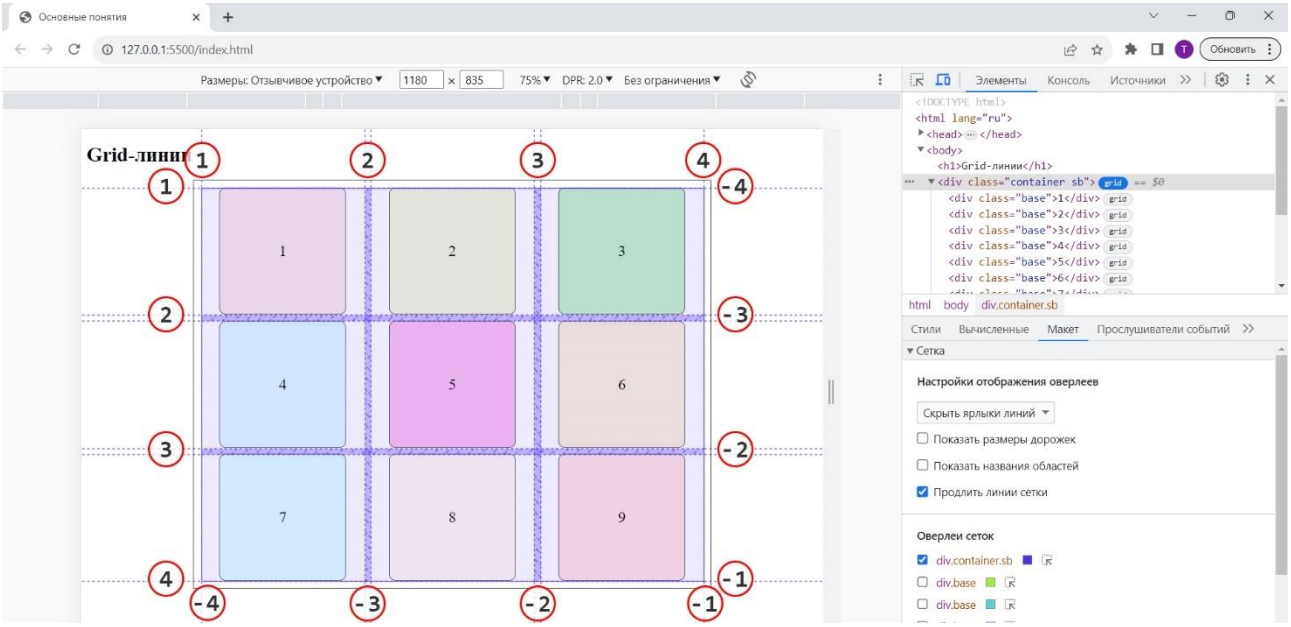
- **Грид-контейнер** — элемент, в котором строится сетка.

- **Грид-элементы** — элементы, размещаемые по сетке. Важно: они должны быть непосредственными потомками грид-контейнера (как и во флексбоксах флекс-элементы должны быть непосредственными потомками флекс-контейнера).

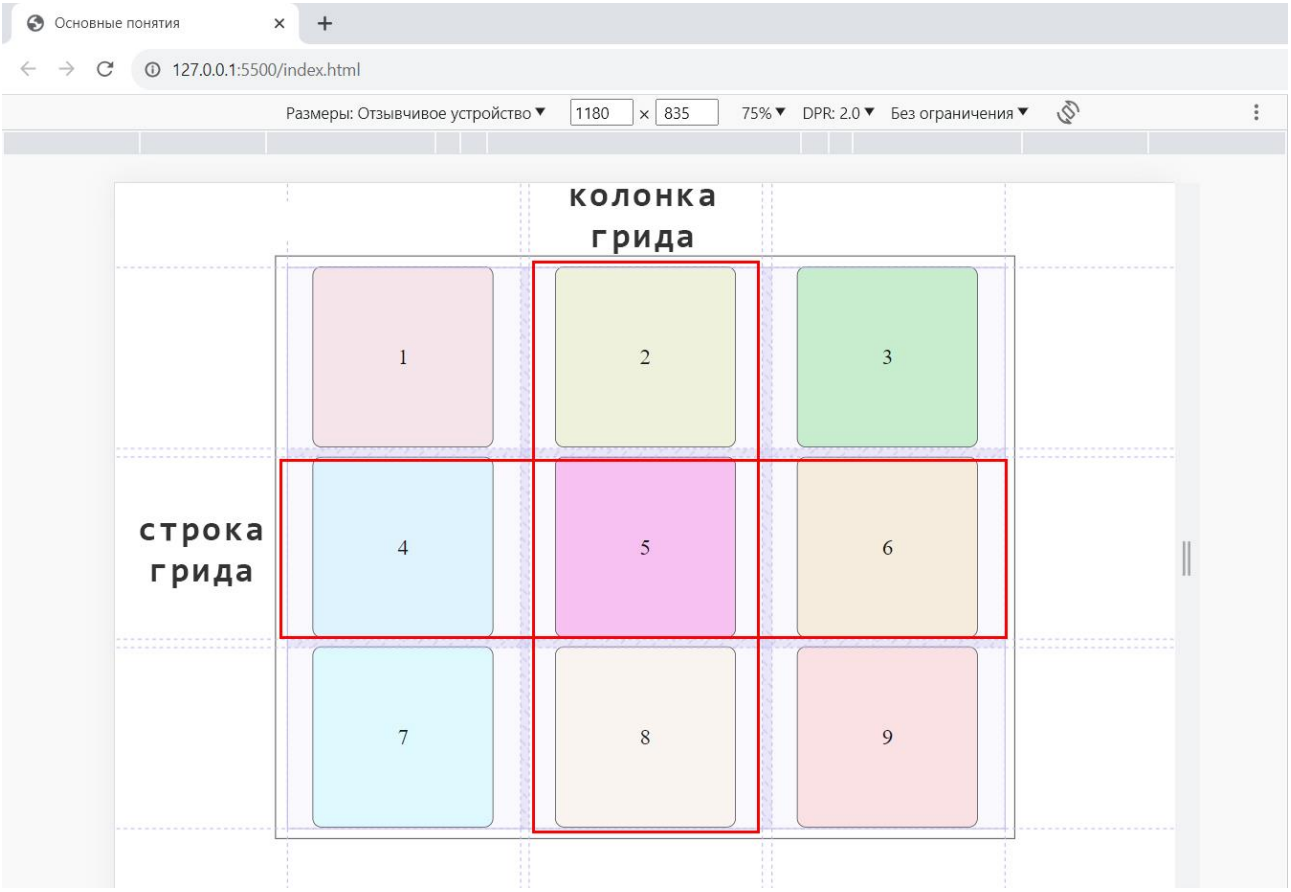


- **Грид-линии** — невидимые вертикальные и горизонтальные линии, разделяющие грид на ячейки и формирующие его структуру. Грид-линии автоматически нумеруются, а также им можно задавать имена (см. раздел лекции).

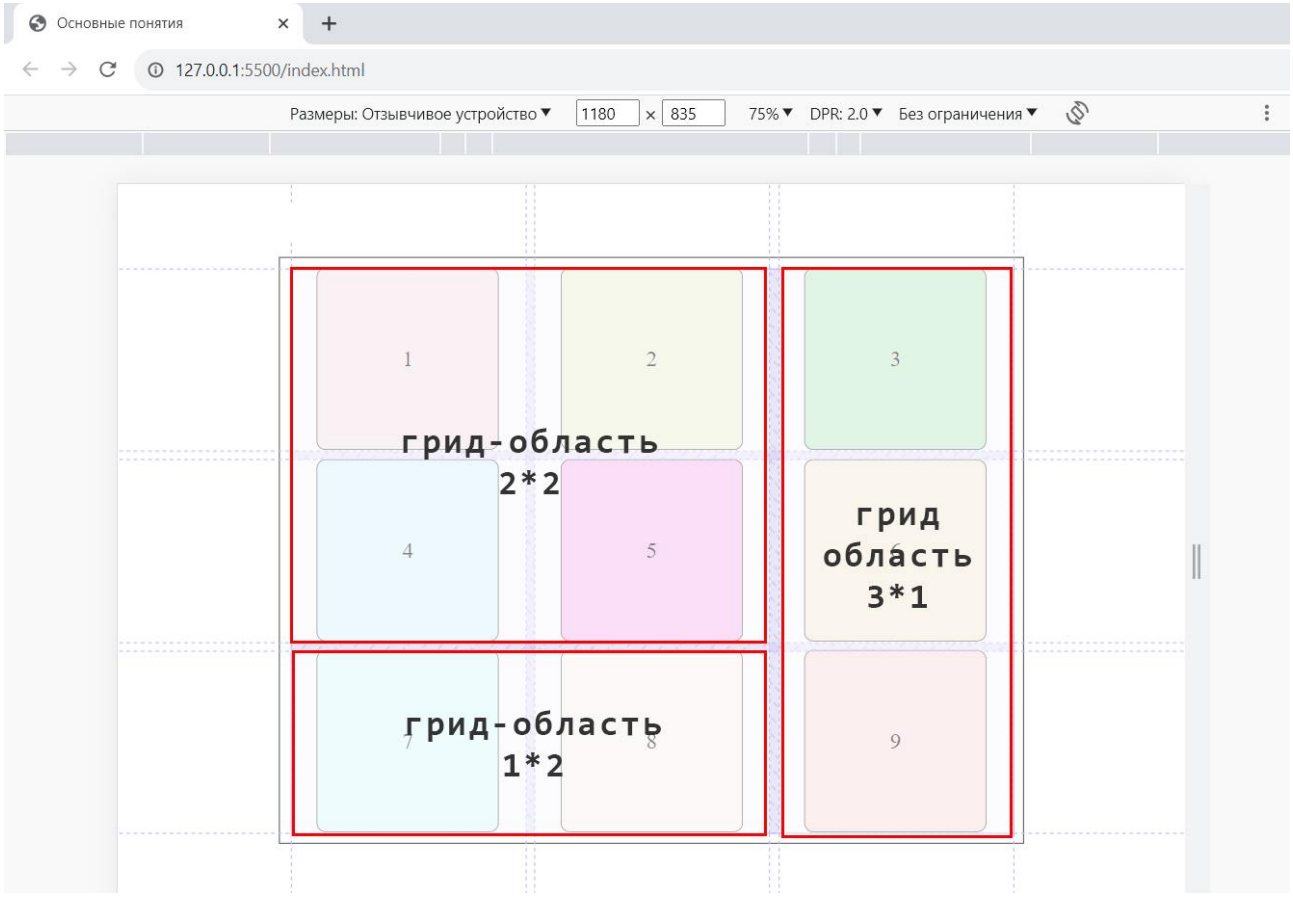
  У каждой грид-линии, именованной или нет, есть два номера: с начала грида, положительное число (1, 2, 3, …), и с конца грида, отрицательное число (-1, −2, −3, …). Привязывать грид-элементы к ним можно как по именам, так и по любому из этих номеров.



- **Грид-полосы** — то, что ограничено парой соседних грид-линий. Вертикальные грид-полосы — это колонки грида (аналог столбцов таблицы), горизонтальные — ряды (аналог строк).
- **Грид-ячейки** — то, что находится на пересечении двух грид-полос (колонки и ряда). Аналог ячейки таблицы.



- **Грид-области** — прямоугольники из M×N смежных грид-ячеек (1×1 и больше). Каждая грид-область ограничена двумя парами грид-линий (парой вертикальных и парой горизонтальных). В них и размещаются грид-элементы. Грид-областям также можно задавать имена.



- **Грид-интервалы** — пустые пространства (отступы) между соседними грид-полосами. Аналог border-spacing в таблице.

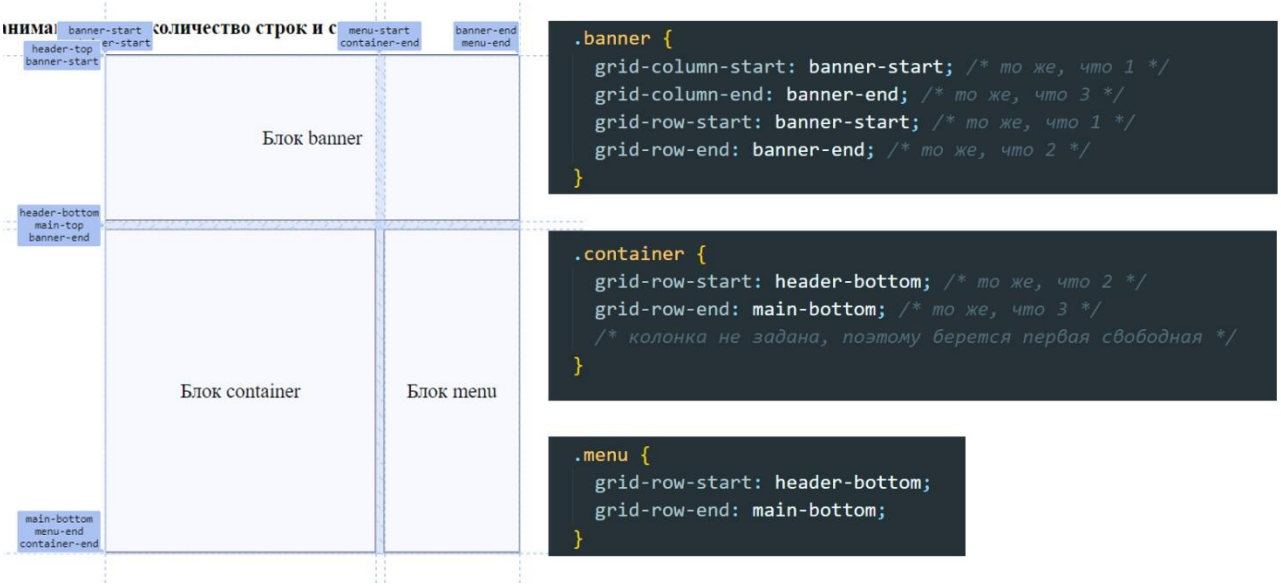
Именование элементов грида

Имена удобны для привязки грид-элементов к ним. Пишутся в квадратных скобках перед или после размера соответствующей полосы.

У одной грид-линии может быть много имен. Имена грид-линий могут быть практически любыми — как имена классов. Допустимы и русские буквы, и другие символы юникода, и

даже эмодзи. Важно соблюсти баланс между «красотой» и читаемостью.

```
.gta-1 {
  grid-template-columns: 2fr 1fr;
  grid-template-rows: [header-top] 200px [header-bottom main-top] 1fr [main-bottom];
  grid-template-areas:
                      "banner    banner"
                      "container menu";
}
```




```
.banner {
  grid-column-start: banner-start; /* то же, что 1 */
  grid-column-end: banner-end; /* то же, что 3 */
  grid-row-start: banner-start; /* то же, что 1 */
  grid-row-end: banner-end; /* то же, что 2 */
}
```

```
.container {
  grid-row-start: header-bottom; /* то же, что 2 */
  grid-row-end: main-bottom; /* то же, что 3 */
  /* колонка не задана, поэтому берется первая свободная */
}
```

```
.menu {
  grid-row-start: header-bottom;
  grid-row-end: main-bottom;
}
```

У имен грид-линий, оканчивающихся на -start и -end, есть еще одна особенность: они косвенно создают именованные грид-области. Пара параллельных грид-линий с именами XXX-start и XXX-end автоматически создает между собой именованную грид-область с именем XXX.

Верно и обратное. При создании грид-области с именем XXX ограничивающие ее грид-линии получают имена XXX-start и XXX-end.

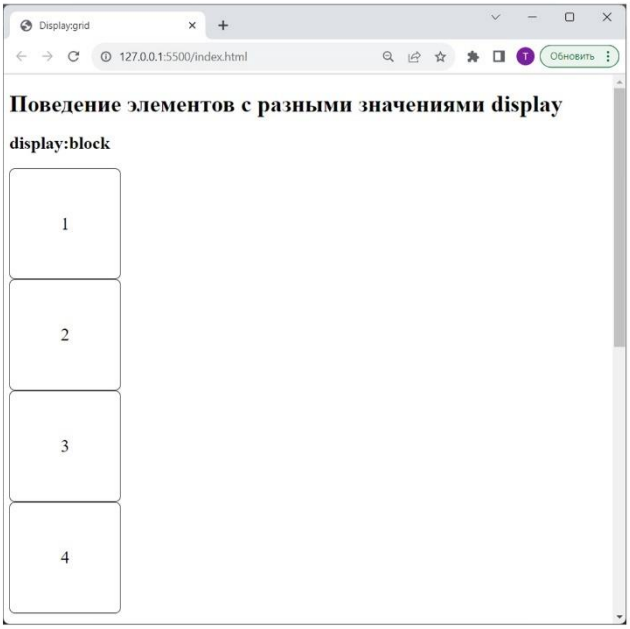
Свойства grid

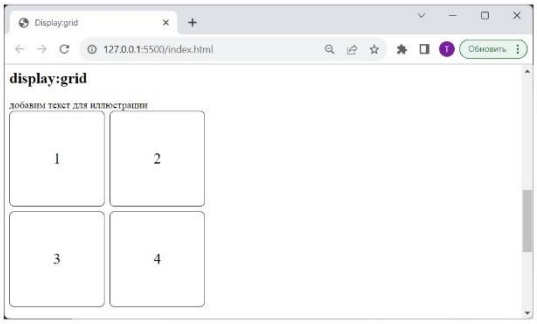
Все свойства можно разделить на две группы

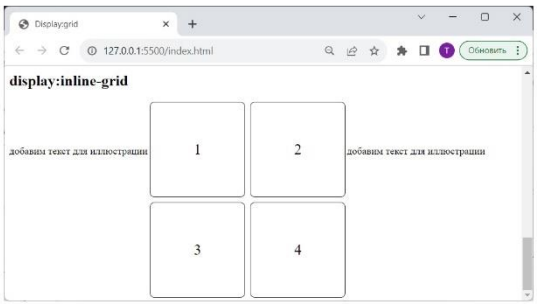
- родительские (свойства grid-контейнера) и
- дочерние (свойства grid-элементов)

Но начинается все со свойства **display** и его значений

- **grid** - элемент ведет себя как блочный, содержимое размещается в соответствии с grid моделью.
- **inline-grid** – элемент ведет себя как строчный, но содержимое размещается в соответствии с grid моделью.
- **subgrid** – если ваш grid элемент является частью сетки, то вы можете использовать это свойство для обозначения того, чтобы размеры строк/колонок были взяты из родительского элемента, а не определяли собственный.







Следующая группа свойств задает **параметры сетки**: размеры, расположение и имена строк и столбцов.

Размеры грид полос можно задавать в любых CSS-единицах длины, абсолютных (pt, mm, px…), относительных (em, rem, vw, vh…), в процентах (от доступной ширины или указанной высоты грид-контейнера).

Интересно, что для задания гибкого размера элементам сетки часто используется единица измерения **fr**. Это дробная единица гибкости. 1fr равна 1 части доступного пространства. Благодаря fr вам больше не нужно пересчитывать ширину

разделов. Единица гибкости fr берет 100% экрана и равномерно распределяет его между элементами сетки.

Значение fr еще называют flex-фактором (flex factor).

Вычисление пропорциональных размеров производится по формуле:

$$\frac{flex - \text{фактор} * \text{доступное пространство}}{\text{сумма всех flex} - \text{факторов}}$$

При этом под доступным пространством понимается **все пространство** grid-контейнера **за исключением фиксированных значений** строк и столбцов.

Также с помощью ключевых слов:

- **min-content —** наименьший возможный размер контента. Для текста это ширина самого длинного неразрываемого фрагмента или слова.
- **max-content —** наибольший размер контента. Для текста это длина самой большой строки без переносов.
- **auto —** размер грид-полосы подстраивается под размеры ее грид-элементов, так, чтобы самый большой из них уместился впритык. Не дает грид-полосе ужиматься меньше min-width самого широкого грид-элемента (для колонки) или min-height самого высокого (для ряда) и растягиваться больше, чем max-content. Но есть важный нюанс: грид-полосы с размером auto могут растягиваться при *-content: stretch (и отсутствии других полос с размерами во fr)

Еще один способ - функция **minmax(<минимум>, <максимум>)**. Задает минимум, до которого можно ужимать полосу, и максимум, на который она может растянуться, если хватит места (аналог min/max-width или min/max-height в одной записи). Максимум можно задавать всеми

вышеперечисленными способами, минимум — всеми, кроме fr. Обычно минимум делают фиксированным, а максимум — гибким, например minmax(200px, 30%) или minmax(5em, 2fr).

Функцией **fit-content()** с максимальным размером в качестве параметра. Если контент не больше этого размера, ведет себя как auto, если больше — этот параметр становится размером полосы.

grid-template

Задать количество и размеры элементов сетки можно обобщающим свойством **grid-template** или по отдельности.

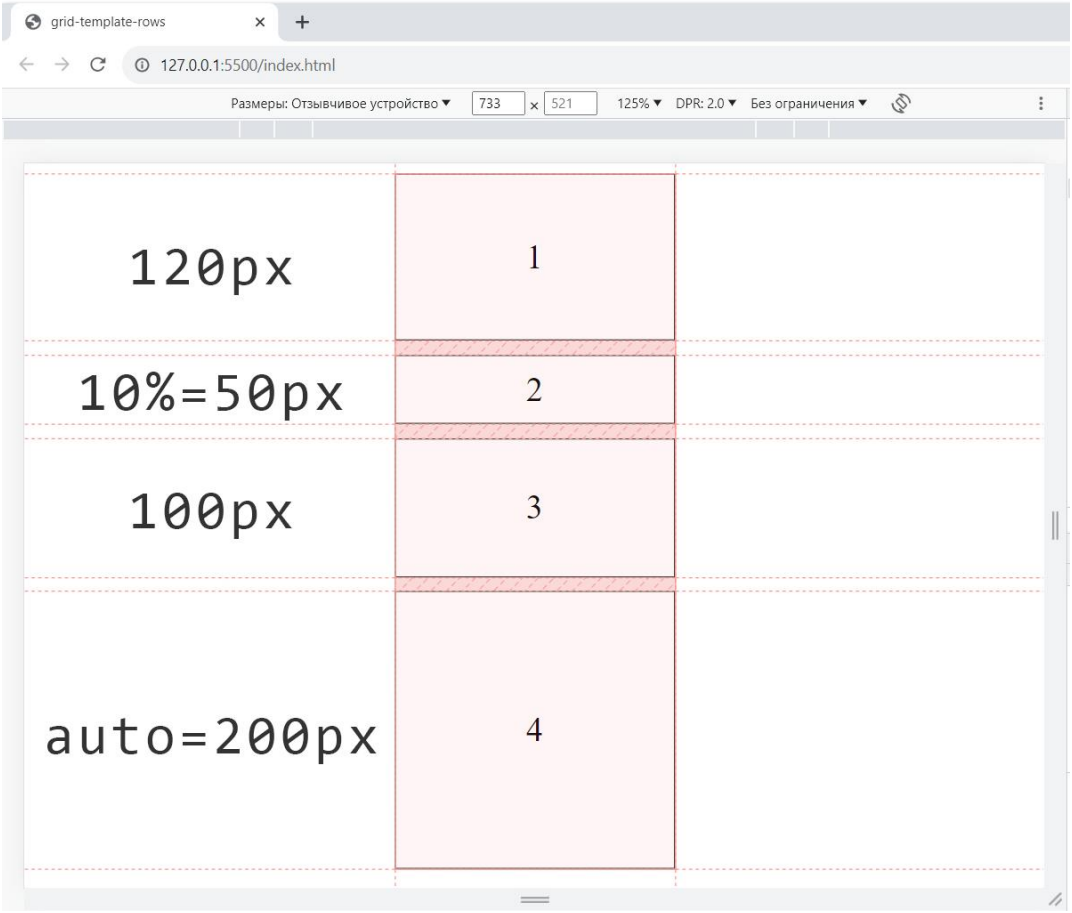
grid-template-columns

Данное свойство используется для определения количества и ширины колонок. При этом, можно определять как свойства для каждой колонки в отдельности, так и устанавливать ширину всех колонок с помощью функции **repeat()**.

grid-template-rows

Данное свойство используется для определения количества и высоты строк. При этом, можно определять как высоту каждой строки в отдельности, так и устанавливать одну для всех строк с помощью функции **repeat()**.

```
.gtr-1{
  height: 500px;
  grid-template-rows:120px 10% 100px auto;
}
```



Объединенная запись будет выглядеть следующим образом

```
grid-template: 20px auto 10% / 1fr 100px auto ;
```

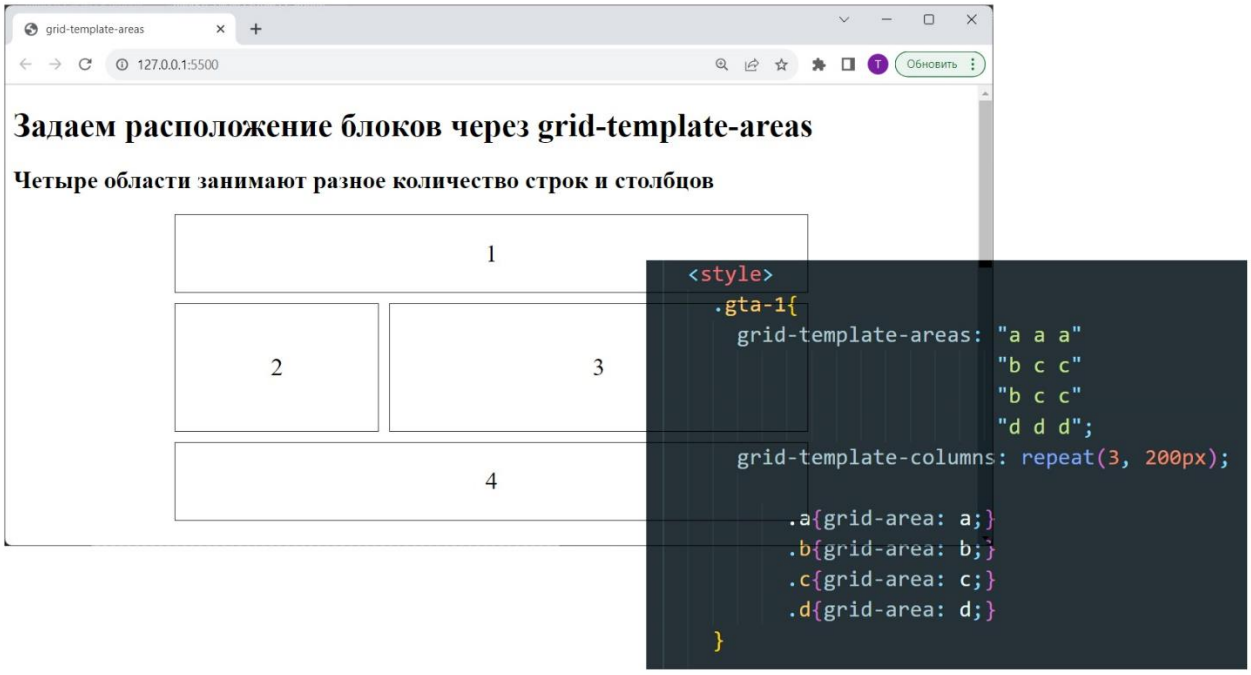
При этом сначала описывают размеры строк грида, а затем столбцов.

grid-template-areas

Данное свойство используется для определения количества пространства, занимаемого ячейкой грида (grid cell), в терминах колонок и строк, в родительском контейнере.

Для получения результата требуется не только родительское, но и хотя бы одно дочернее свойство:

- **grid-template-areas**: родительское свойство, создающее схему расположения ячеек;
- **grid-area**: дочернее свойство, которое использует схему и указывает какой из блоков разметки является каким элементом схемы.

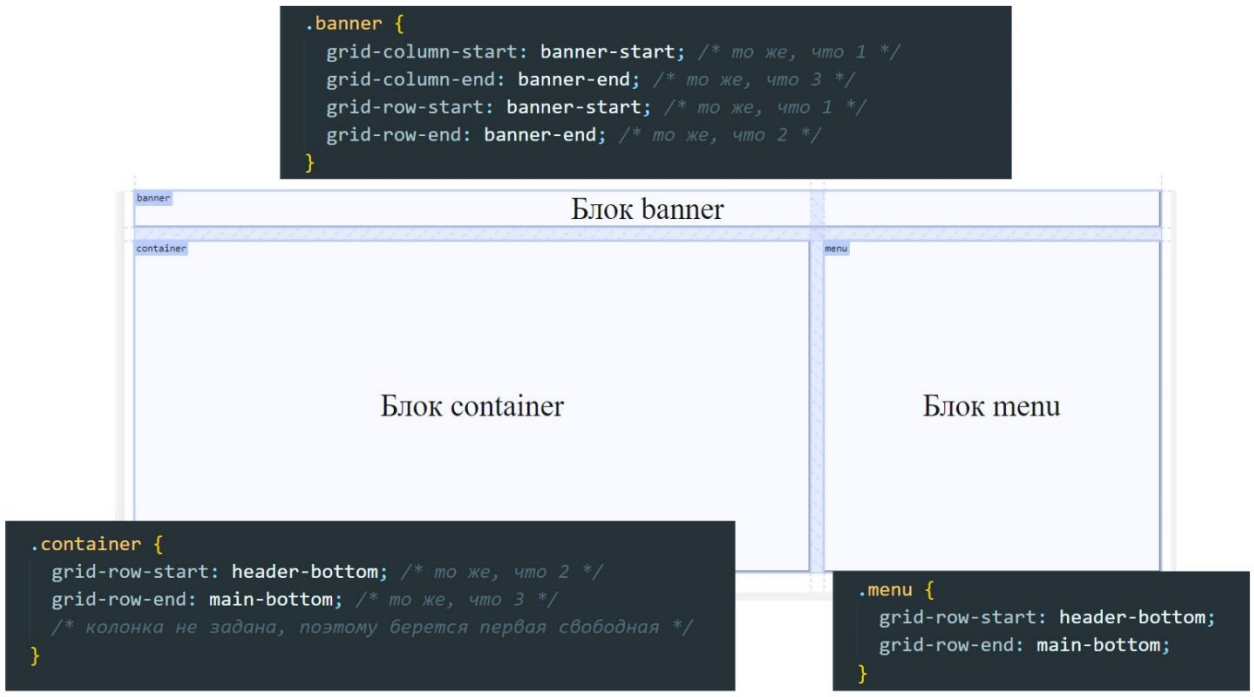


Если одну из ячеек грида нужно пропустить, укажем на ее месте точку (.).

Как уже было сказано при создании грид-области с именем XXX ограничивающие ее грид-линии получают имена XXX-start и XXX-end.

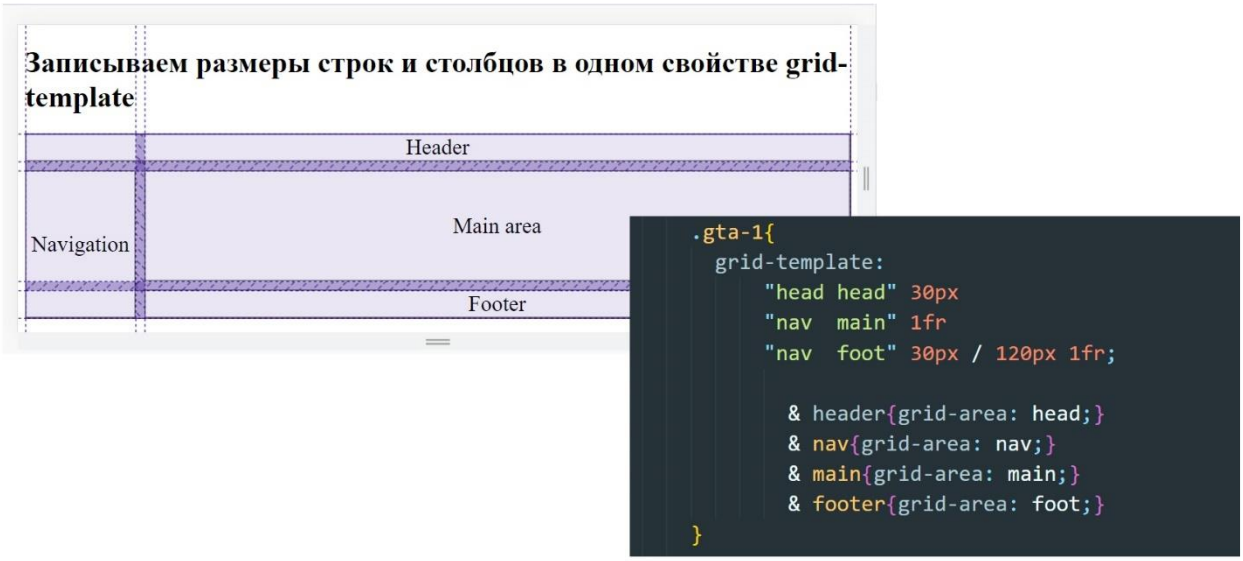
Мы можем использовать одновременно и размеры, и grid-template-areas для определения области.

```
.gta-1 {
  grid-template-columns: 2fr 1fr;
  grid-template-rows: [header-top] auto [header-bottom main-
top] 1fr [main-bottom];
  grid-template-areas:
                    "banner    banner"
                    "container menu";
      }
```



grid-template

При объединенной записи, сначала указываем количество строк, затем столбцов или сначала область, затем размеры строк и столбцов.



column-gap, row-gap, gap

Свойство **gap** задаёт отступы между столбцами и строками, а не вдоль края контейнера сетки. Является сокращением для свойств **row-gap** и **column-gap**.

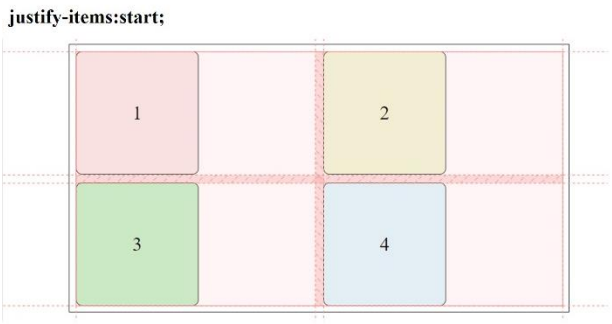
Одно значение, указанное в свойстве **gap** задает одинаковые отступы и между строками, и между столбцами. Два значения задают расстояние между строками – первое, между столбцами – второе. В качестве значения также может быть использовано вычисляемое с помощью **calc()** значение.

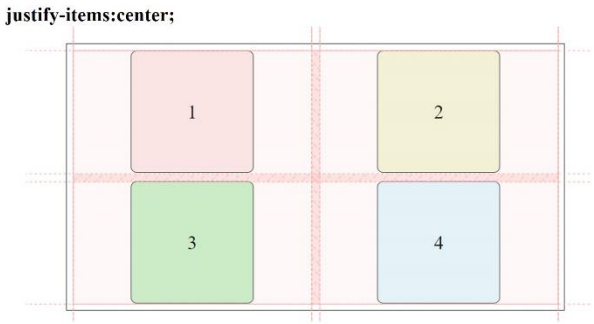
justify-items

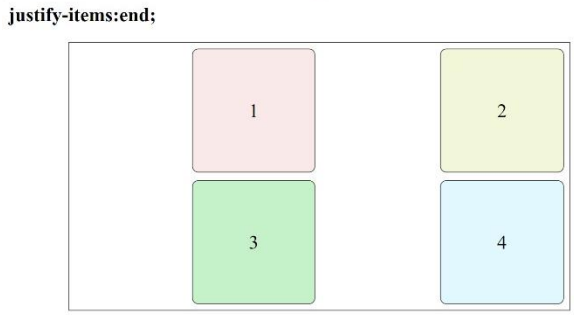
Данное свойство используется для позиционирования grid-элементов внутри grid-контейнера вдоль главной оси. Оно принимает одно из 4 возможных значений, аналогично CSS Flexbox.

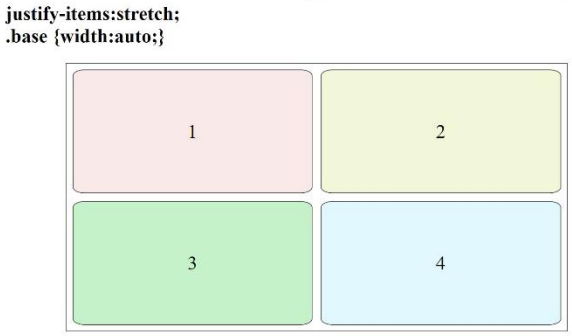
- **start** располагает содержимое в начале главной оси.
- **end** располагает содержимое в конце главной оси.

- **center** располагает содержимое по центру главной оси.
- **stretch** располагает на всю ширину области (по умолчанию).





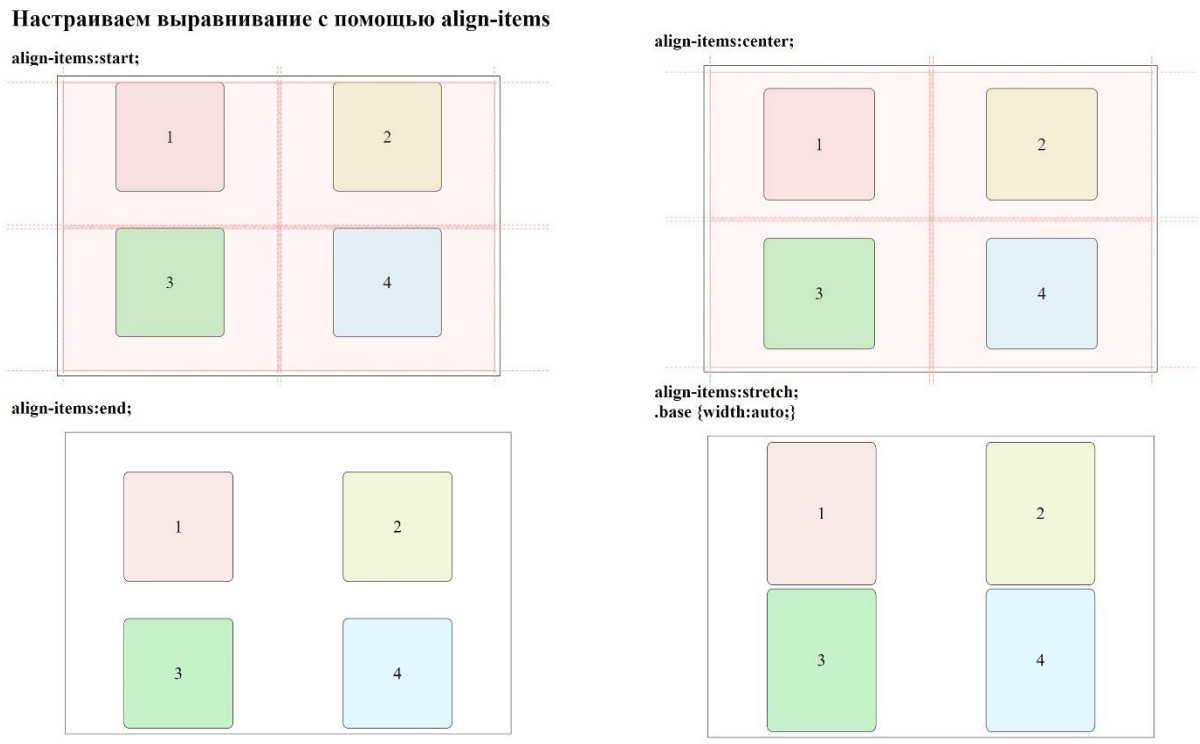




align-items

Данное свойство используется для позиционирования grid-элементов внутри grid-контейнера вдоль поперечной оси. Оно принимает одно из 4 возможных значений:

- **start** располагает содержимое в начале поперечной оси.
- **end** располагает содержимое в конце поперечной оси.
- **center** располагает содержимое по центру поперечной оси.
- **stretch** располагает на всю высоту области (по умолчанию).

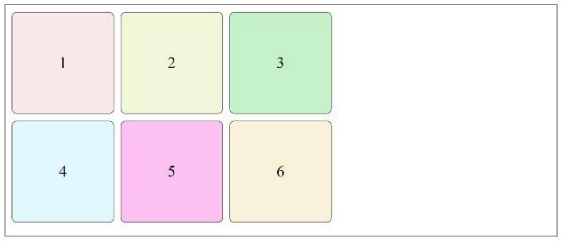


justify-content

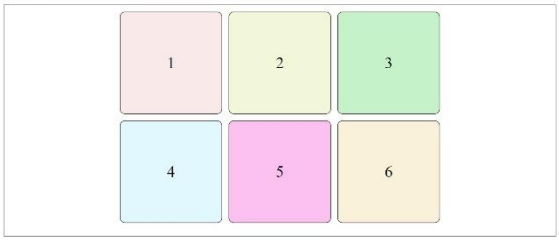
Данное свойство используется для позиционирования сетки целиком внутри grid-контейнера **вдоль основной оси**. Оно принимает одно из 7 возможных значений:

- **start** выравнивает сетку по левой стороне контейнера.
- **end** выравнивает сетку по правой стороне контейнера.
- **center** выравнивает сетку по центру контейнера.
- **stretch** масштабирует элементы чтобы сетка могла заполнить всю ширину контейнера.
- **space-around** одинаковое пространство между элементами, и полуразмерные отступы по краям.
- **space-between** одинаковое пространство между элементами, без отступов по краям.
- **space-evenly** одинаковое пространство между элементами, и полноразмерные отступы по краям.

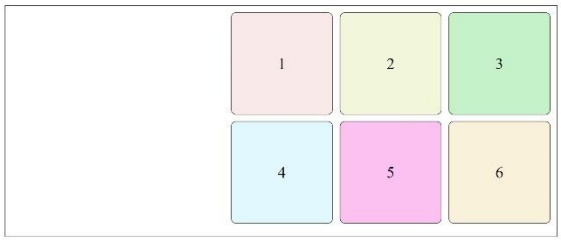
justify-content:start;



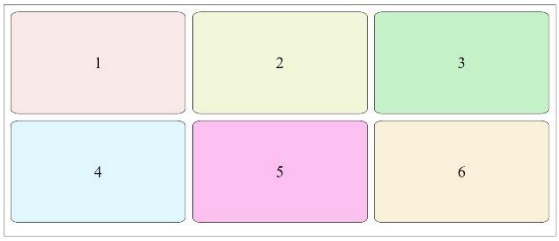
justify-content:center;



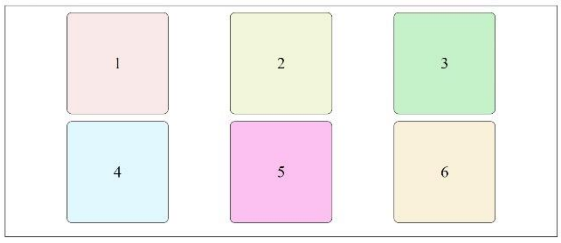
justify-content:end;



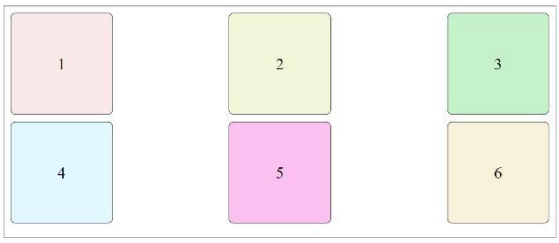
justify-content:stretch;



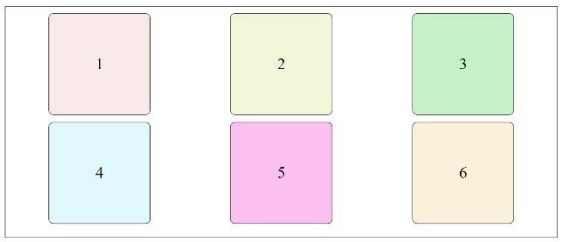
justify-content:space-evenly;



ustify-content:space-between;



justify-content:space-around;



align-content

Данное свойство используется для позиционирования сетки внутри грид-контейнера **вдоль поперечной оси**. Оно принимает 7 возможных значений:

- **start** выравнивает сетку по верхней части контейнера.
- **end** выравнивает сетку по нижней части контейнера.
- **center** масштабирует элементы чтобы сетка могла заполнить всю высоту контейнера.
- **space-around** одинаковое пространство между элементами, и полуразмерные отступы по краям.

- **space-between** одинаковое пространство между элементами, без отступов по краям.
- **space-evenly** одинаковое пространство между элементами, и полноразмерные отступы по краям.

justify-self | align-self

Justify-self выравнивает содержимое элемента вдоль оси строки (в отличии от align-self, который выравнивает вдоль оси столбца).

Применяется к: содержимому внутри отдельного элемента.

- **start** Выравнивает содержимое по началу оси.
- **end** Выравнивает содержимое в конце оси.
- **center** Выравнивает содержимое по центру области.
- **stretch** Заполняет всю ширину|высоту области (по умолчанию).

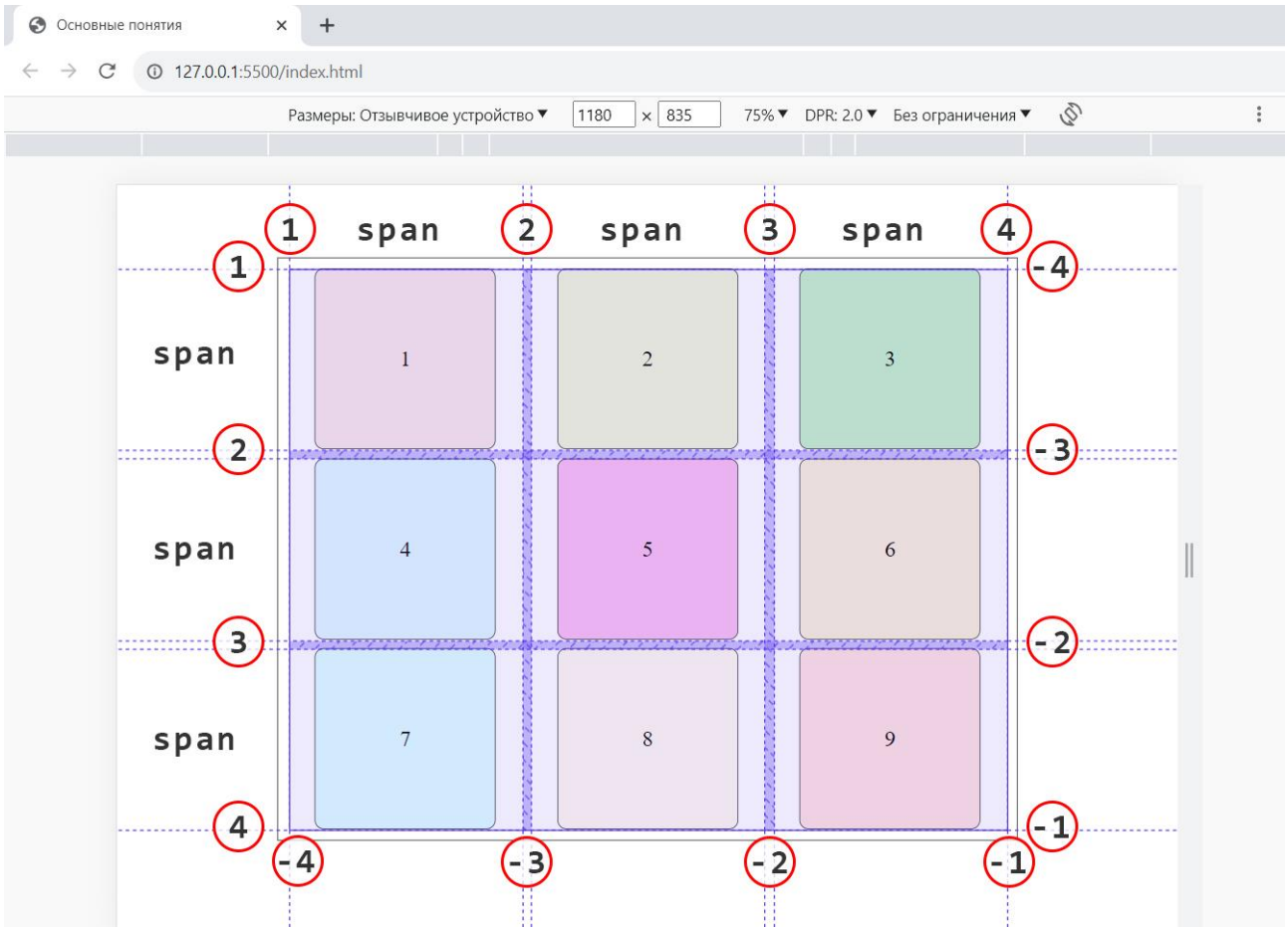
Объединение элементов

В отличие от flexbox-ов, работая с grid-ами мы можем объединять содержимое сетки.

Для вычисления строк и колонок при их объединении используется два вида единиц:

- целые числа (1, 2, 3 и т.д.)
- ключевое слово span

Целые числа указывают на границы блоков, а span-ы на сами блоки.



grid-column: start/end

Данное свойство позволяет объединять колонки. Оно является сокращением для:

- **grid-column-start** – начало объединенной колонки
- **grid-column-end** – конец объединенной колонки

Например, размеры блока 1 с предыдущего рисунка можно описать следующими **с**пособами:

1 способ, напрямую задаем начальную и конечную линии сетки

```
    grid-column-start: 1;
    grid-column-end: 2;
```

Сокращаем запись, разделяем начало и конец косой чертой

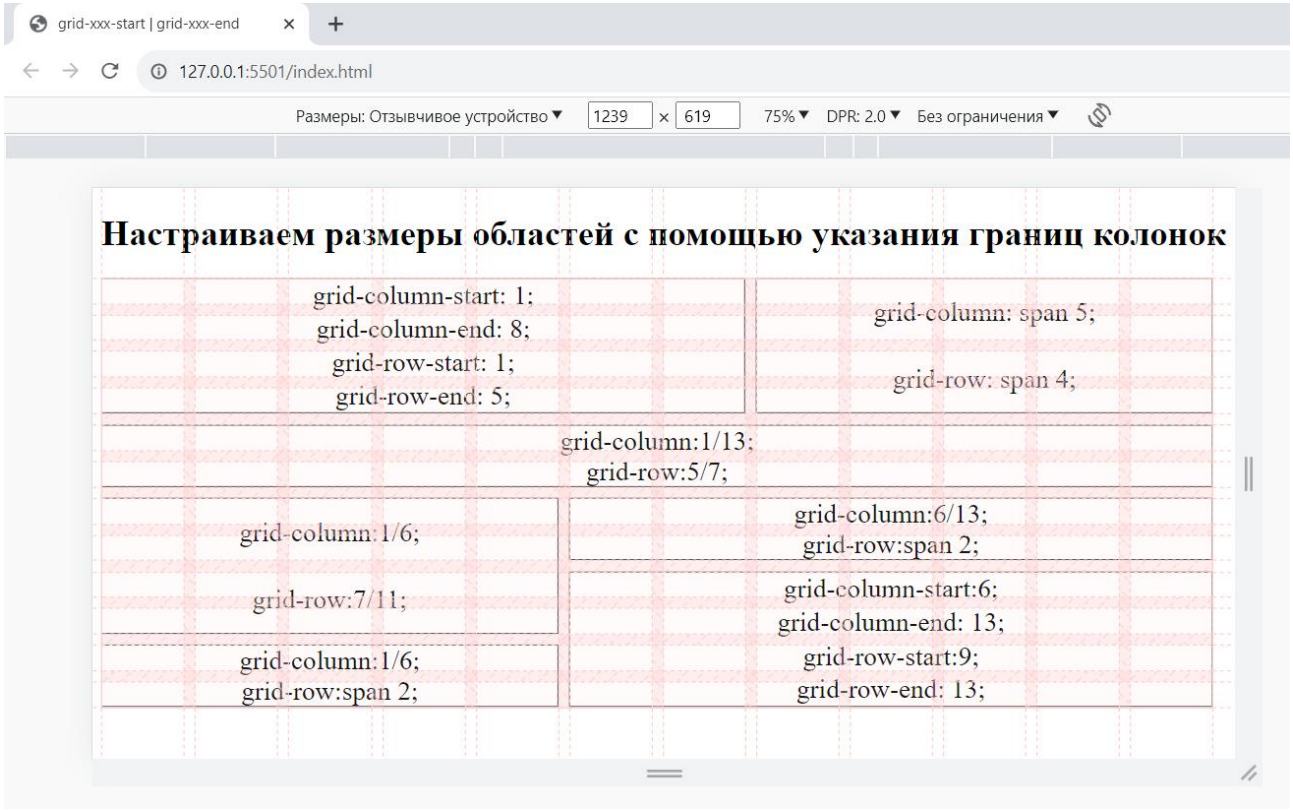
```
    grid-column: 1 / 2;
```

Указываем, что блок будет занимать размеру одного пространства между границами

```
    grid-column: span 1;
```

Свойство **grid-row: start/end** работает аналогично и позволяет объединять строки. Является сокращением для:

- grid-row-start
- grid-row-end



Можно привязывать элементы к грид-линиям **по номеру** (любому из двух — положительному, с начала, или отрицательному, с конца), **по имени**, **по имени и номеру** (если одноименных линий несколько, например, при repeat).

```
/* 3 колонки до предпоследней включительно */
grid-column: span 3 / -2;

/* от конца области menu до 2-й линии с именем "item-column" */
grid-column: menu-end / span item-column 2;

/* от второй по счету линии перед конечной до пятой линии */
grid-row-start: span 2; grid-row-end: 5;
```

```
/*  поставить элемент между 5-й и 2-й линиями колонок с конца
(т.е. он займет 3 колонки, 4-ю от конца явного грида, 3-ю от конца и
предпоследнюю) */
grid-column-start: -5; grid-column-end: -2;

/* растянуть элемент на всю ширину грида, от начала до конца, можно так
*/
grid-column-start: 1; grid-column-end: -1;

/* означает «охватить ряды вплоть до линии с именем sidebar»
    (если таких линий несколько – то до ближайшей из них) */
grid-row-end: span sidebar;
```

При повторяющейся структуре грида может быть несколько линий с одинаковым именем. Чтобы привязаться к одной конкретной из них, указываем и имя, и номер:

grid-column-start: content 3 — началом элемента будет **3-я** по счету линия с именем **content**.

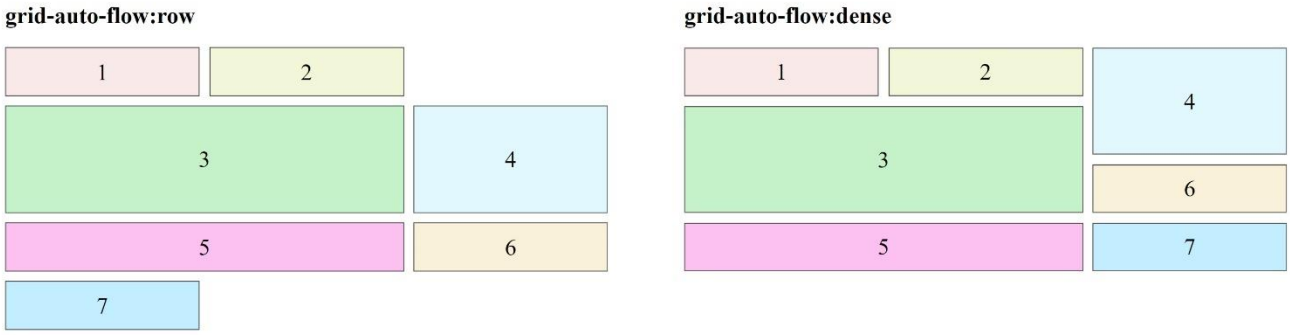
grid-auto-flow

Свойство **grid-auto-flow** управляет тем, как работает алгоритм автоматического размещения, точно указывая, как элементы попадают в сетку.

Значение по умолчанию: row.

- **row** Алгоритм размещает элементы, заполняя каждую строку поочередно, добавляя новые строки по мере необходимости.
- **column** Алгоритм помещает элементы, заполняя каждый столбец поочередно, добавляя по мере необходимости новые столбцы.
- **dense** алгоритм использует «плотный» алгоритм упаковки, который пытается заполнить дыры в сетке, если позже появятся более мелкие элементы. Это может привести к тому, что элементы появятся не по порядку, но

при этом заполнят отверстия, оставленные более крупными элементами.



Сокращенные свойства

place-content

Данное свойство является сокращением для: **align-content** и **justify-content**. Позиционируем grid внутри сетки.

place-items

Позволяет одновременно выравнивать элементы и в колонки, и в строки (т.е. по свойствам **align-items** и **justify-items**) в соответствующей системе раскладки, такой как grid или flexbox. Если задано одно значение, оно используется для выравнивания и в колонке, и в строке.

place-self

Данное свойство является сокращением для: **align-self** и **justify-self.**

ЗАДАНИЕ № 34. Верстка по макету.

В задании представлены несколько примеров макетов страниц. Реализуйте их с помощью CSS Grid. Тематика страниц на выбор.

Используйте для каждого макета свой подход: именованные области, именованные линии, простую нумерацию элементов сетки.

Важно. Варианты реализации к заданию не приложены, т.к. единственно верного решения нет, а ограничивать возможные варианты своим видением не хочется. Смотрите примеры кода.

Макет 1.

<table>
<tr><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td colspan="4"></td></tr>
<tr><td rowspan="3"></td><td rowspan="2"></td><td colspan="2"></td></tr>
<tr><td rowspan="3"></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="4"></td></tr>
</table>

Макет 2.

<table>
<tr><td colspan="4"></td></tr>
<tr><td colspan="4"></td></tr>
<tr><td rowspan="6"></td><td rowspan="2"></td><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td rowspan="2"></td><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td rowspan="2"></td><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="4"></td></tr>
</table>