display

Многоцелевое свойство, которое определяет, как элемент должен быть показан в документе.

Применяется ко: всем элементам.

1. grid формирует сетку как блок.
2. inline-grid формирует сетку как строчный блок.
3. subgrid если ваш контейнер это ещё и элемент (вложенная сетка), то вы можете использовать это свойство для обозначения того, чтобы размеры строк/колонок были взяты из родительского элемента, а не определяли собственный.

justify-items

Выравнивает содержимое вдоль оси строки (в отличии от align-items который выравнивает элементы вдоль оси столбца). Это значение применяется ко всем элементам сетки внутри контейнера.

Наследуется: нет.

Применяется ко: всем элементам.

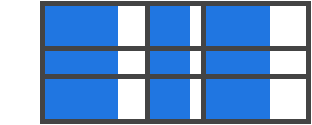
Значение по умолчанию: legacy.

1. start выравнивает содержимое по левой стороне области.
2. end выравнивает содержимое по правой стороне области.
3. center выравнивает содержимое по центру области.
4. stretch заполняет всю ширину области (по умолчанию).

.parent {

display: grid;

justify-items: start; }-



align-items

Выравнивает содержимое вдоль оси столбца (в отличии от justify-items который выравнивает элементы вдоль оси строки). Это значение применяется ко всем элементам сетки внутри контейнера.

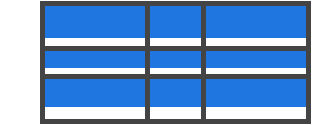
Наследуется: нет.

Применяется ко: всем элементам.

Значение по умолчанию: normal.

1. start выравнивание элементов по верхней части области.
2. end выравнивание элементов по нижней части области.
3. center выравнивание элементов по центру области.
4. stretch заполняет всю высоту области (по умолчанию).

.parent { display: grid;  
 align-items: center;  
align-items: start;}



justify-content

Выравнивает содержимое вдоль оси столбца (в отличии от justify-items который выравнивает элементы вдоль оси строки). Это значение применяется ко всем элементам сетки внутри контейнера.

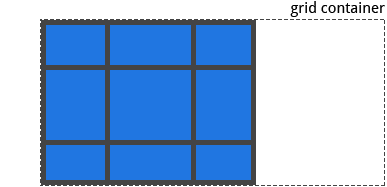
Наследуется: нет.

Применяется ко: всем элементам.

Значение по умолчанию: normal.

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

.parent {  
 display: grid;  
 justify-content: stretch;  
justify-content: start; }



align-content

Это свойство выравнивает сетку вдоль оси колонки (в отличии от свойства justify-content, которое выравнивает сетку вдоль оси строки).

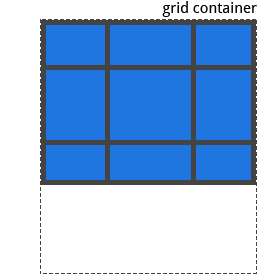
Наследуется: нет.

Применяется к: block контейнерам, flex контейнерам и grid контейнерам.

Значение по умолчанию: normal.

1. start выравнивает сетку по верхней части контейнера.
2. end выравнивает сетку по нижней части контейнера.
3. center масштабирует элементы чтобы сетка могла заполнить всю высоту контейнера.
4. space-around одинаковое пространство между элементами, и полуразмерные отступы по краям.
5. space-between одинаковое пространство между элементами, без отступов по краям.
6. space-evenly одинаковое пространство между элементами, и полноразмерные отступы по краям.

.parent { display: grid;  
 align-content: space-evenly;  
align-content: start;}



grid-auto-columns grid-auto-rows

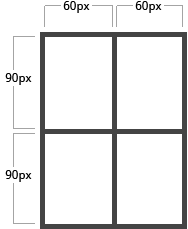
Определяет размер любых автоматически созданных треков (иначе говоря, неявных треков). Неявные треки создаются при явном позиционировании столбцов и строк (через grid-template-rows/grid-template-columns), которые находятся за пределами заданной сетки.

Значения:

<track-size> - может быть фиксированным размером, процентами, или частью свободного пространства в сетке (с использованием единицы fr);

.container {  
 grid-auto-columns: <track-size> ...;  
 grid-auto-rows: <track-size> ...; }

Чтобы проиллюстрировать то как создаются неявные треки, посмотрите на это:



Пример выше создаст сетку 2 x 2.

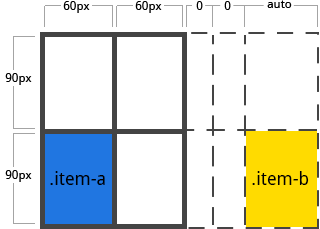
А теперь представьте, что вы позиционируете элементы сетки с помощью grid-column и grid-row следующим образом:

.item-a {  
 grid-column: 1 / 2;  
 grid-row: 2 / 3; }

.item-b {  
 grid-column: 5 / 6;  
 grid-row: 2 / 3; }

Мы сказали чтобы элемент .item-b начинался с 5 линии и заканчивался на 6 линии, по оси столбцов, но мы не определяли линии 5 и 6. Так как мы ссылаемся на не существующие линии, неявные треки с шириной 0 были созданы чтобы заполнить пробелы. Мы можем использовать grid-auto-columns и grid-auto-rows чтобы указать ширину этим неявным трекам:

.container {  
 grid-auto-columns: 60px; }



***grid-auto-flow***

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

Наследуется: нет.

Применяется к: grid контейнерам.

Значение по умолчанию: row.

1. row Алгоритм размещает элементы, заполняя каждую строку поочередно, добавляя новые строки по мере необходимости.
2. column Алгоритм помещает элементы, заполняя каждый столбец поочередно, добавляя по мере необходимости новые столбцы.
3. 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.

.parent {  
  display: grid;  
  grid-auto-flow: column;  
}

.container {  
 display: grid;  
 grid-template-columns: 60px 60px 60px 60px 60px;  
 grid-template-rows: 30px 30px;  
 grid-auto-flow: row;}

.item-a {  
 grid-column: 1;  
 grid-row: 1 / 3;}

.item-e {  
 grid-column: 5;  
 grid-row: 1 / 3;}

**grid**

Сокращение для настройки всех следующих свойств: grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns, и grid-auto-flow. Он также настраивает grid-column-gap и grid-row-gap на их начальные значения, даже если они не могут быть явно заданы этим свойством.

Значения: none - настраивает все свойства на их начальные значения;

<grid-template-rows> / <grid-template-columns> - настраивает grid-template-rows и grid-template-columns на соответствующие указанные значения, а все остальные свойства сбрасывает к их начальным значениям;

<grid-auto-flow> [<grid-auto-rows> [ / <grid-auto-columns>] ] - принимает те же самые значения что и grid-auto-flow, grid-auto-rows и grid-auto-columns, соответственно. Если grid-auto-columns опущено, устанавливается значение соответствующие для grid-auto-rows. Если опущены оба, то они устанавливаются в начальные значения;

.container {  
 grid: none | <grid-template-rows> / <grid-template-columns> | <grid-auto-flow> [<grid-auto-rows> [/ <grid-auto-columns>]]; }

Примеры: Следующие два блока кода эквивалентны:

.container {  
 grid: 200px auto / 1fr auto 1fr; }

.container {  
 grid-template-rows: 200px auto;  
 grid-template-columns: 1fr auto 1fr;  
 grid-template-areas: none; }

И следующие два блока также эквивалентны:

.container {  
 grid: column 1fr / auto; }

.container {  
 grid-auto-flow: column;  
 grid-auto-rows: 1fr;  
 grid-auto-columns: auto; }

Он также принимает более сложный, но довольно удобный синтаксис для настройки всего и сразу. Вы указываете grid-template-areas, grid-template-rows и grid-template-columns, а все остальные свойства устанавливаются в начальные значения. Всё что вы делаете, это указываете имена строк и размеры треков в строке соответствующим областям сетки. Это проще всего описать на примере:

.container {  
 grid: [row1-start] "header header header" 1fr [row1-end]  
[row2-start] "footer footer footer" 25px [row2-end]  
/ auto 50px auto; }

Это эквивалентно следующему:

.container {  
 grid-template-areas:   
"header header header"  
"footer footer footer";  
grid-template-rows: [row1-start] 1fr [row1-end row2-start] 25px [row2-end];  
grid-template-columns: auto 50px auto; }

## Свойства для дочерних элементов (Grid элементы)

#### **grid-column-start grid-column-end grid-row-start grid-row-end**

Определяют местоположение в сетке ссылаясь на конкретные линии. grid-column-start/grid-row-start - это линия с которой начинается элемент, а grid-column-end/grid-row-end - это линия на которой элемент заканчивается.

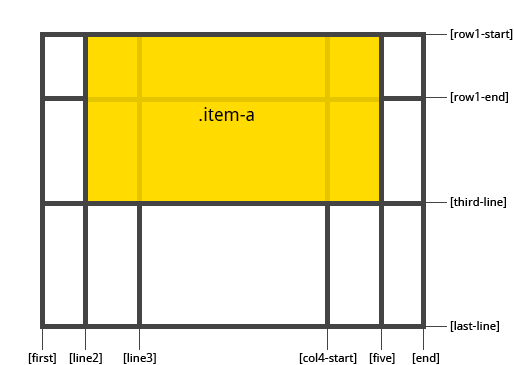
Значения:

* **<line>** - может быть числом ссылающимся на пронумерованную линию, или названием ссылающимся на именованую линию;
* **span <number>** - элемент, который будет охватывать предоставленное количество треков;
* **span <name>** - элемент будет будет охватывать пока не достигнет линии с указанным названием;
* **auto** - указывает автоматическое размещения, автоматическое охват, или охват по умолчанию;

.item {  
 grid-column-start: <number> | <name> | span <number> | span <name> | auto  
 grid-column-end: <number> | <name> | span <number> | span <name> | auto  
 grid-row-start: <number> | <name> | span <number> | span <name> | auto  
 grid-row-end: <number> | <name> | span <number> | span <name> | auto }

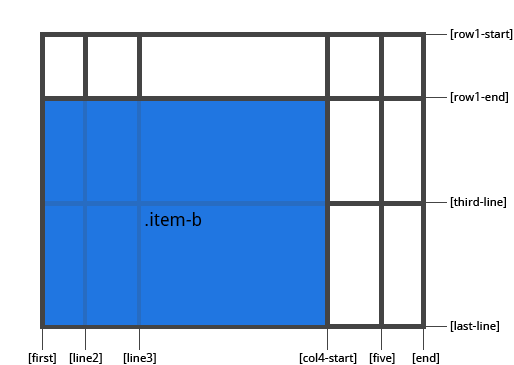
Примеры:

.item-a {  
 grid-column-start: 2;  
 grid-column-end: five;  
 grid-row-start: row1-start  
 grid-row-end: 3 }



.item-b {  
 grid-column-start: 1;  
 grid-column-end: span col4-start;  
 grid-row-start: 2  
 grid-row-end: span 2 }

Если grid-column-end/grid-row-end не объявлены, элемент будет охватывать 1 трек по умолчанию.



#### **grid-column grid-row**

Сокращение для [grid-column-start](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-column-row-start-end) + [grid-column-end](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-column-row-start-end), и [grid-row-start](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-column-row-start-end) + [grid-row-end](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-column-row-start-end), соответственно.

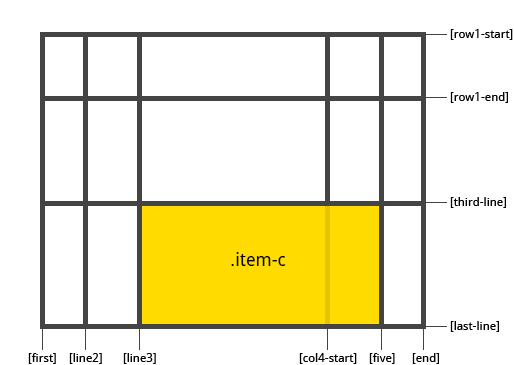
Значения:

* **<start-line> / <end-line>** - каждый из них принимает тоже самое, что и в длинной версии, включая охват;

.item {  
 grid-column: <start-line> / <end-line> | <start-line> / span <value>;  
 grid-row: <start-line> / <end-line> | <start-line> / span <value>; }

Пример:

.item-c {  
 grid-column: 3 / span 2;  
 grid-row: third-line / 4; }



Если значение конечной линии не указано, то элемент будет охватывать только 1 трек, по умолчанию.

#### **grid-area**

Даёт название элементу чтобы можно было ссылаться на него с помощью шаблона созданного через [grid-template-areas](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-template-areas) свойство. В качестве альтернативы, это свойство может быть использовано в качестве сокращения для [grid-row-start](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-column-row-start-end) + [grid-column-start](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-column-row-start-end) + [grid-row-end](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-column-row-start-end) + [grid-column-end](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-column-row-start-end).

Значения:

* **<name>** - название, которое вы выберите;
* **<row-start> / <column-start> / <row-end> / <column-end>** - может быть нумерацией или названиями линий;

.item {  
 grid-area: <name> | <row-start> / <column-start> / <row-end> / <column-end>; }

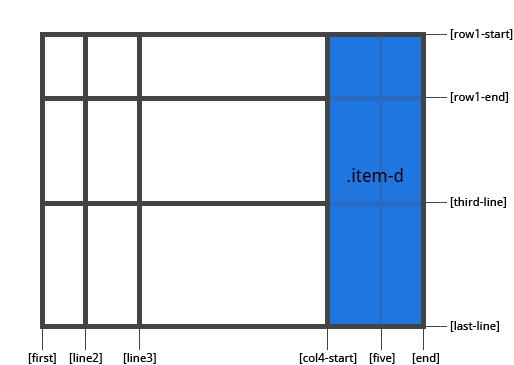
Примеры:

Как способ назначить название элементу:

.item-d {  
 grid-area: header }

Как сокращение для grid-row-start + grid-column-start + grid-row-end+ grid-column-end:

.item-d {  
 grid-area: 1 / col4-start / last-line / 6 }



**justify-self**

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

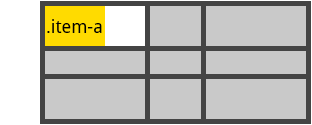
Наследуется: нет.

Применяется к: содержимому внутри отдельного элемента.

Значение по умолчанию: auto.

1. start Выравнивает содержимое по левой части области.
2. end Выравнивает содержимое по правой части области.
3. center Выравнивает содержимое по центру области.
4. stretch Заполняет всю ширину области (по умолчанию).

.parent {  display: grid; }  
.child {   justify-self: end; }



align-self

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

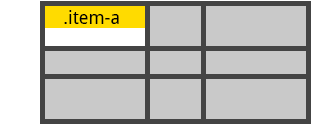
Наследуется: нет.

Применяется к: содержимому внутри отдельного элемента.

Значение по умолчанию: auto.

1. start Выравнивает содержимое по верхней части области.
2. end Выравнивает содержимое по нижней части области.
3. center Выравнивает содержимое по центру области.
4. stretch Заполняет всю высоту области (по умолчанию).

.parent {  display: grid; }  
.child {  align-self: start; }



display: grid;  
grid-template-columns: 150px 1fr 2fr 150px;

.grid\_conteiner {  
display: grid;  
grid-template-columns: 150px minmax(20%, 2fr) minmax(40%, 300px) 150px;

.grid\_conteiner {  
display: grid;  
grid-template-columns: 150px minmax(20%, 2fr) minmax(40%, 300px) 150px; // 4 колонки  
grid-template-rows:minmax(100px, 1fr) 150px 200px; // 3 ряда

.container {  
 grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];  
 grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line]; }

.container{  
 grid-template-rows: [row1-start] 25% [row1-end row2-start] 25% [row2-end];}  
Если в вашем определении содержатся повторяющиеся части, то можно использовать нотацию repeat():

.container {  
 grid-template-columns: repeat(3, 20px [col-start]) 5%;}

Тоже самое что и:

.container {  
 grid-template-columns: 20px [col-start] 20px [col-start] 20px [col-start] 5%; }

Единица fr позволяет вам настраивать размер треков как часть свободного пространства в контейнере. Вот пример, который устанавливает каждому элементу одну третью ширины контейнера.

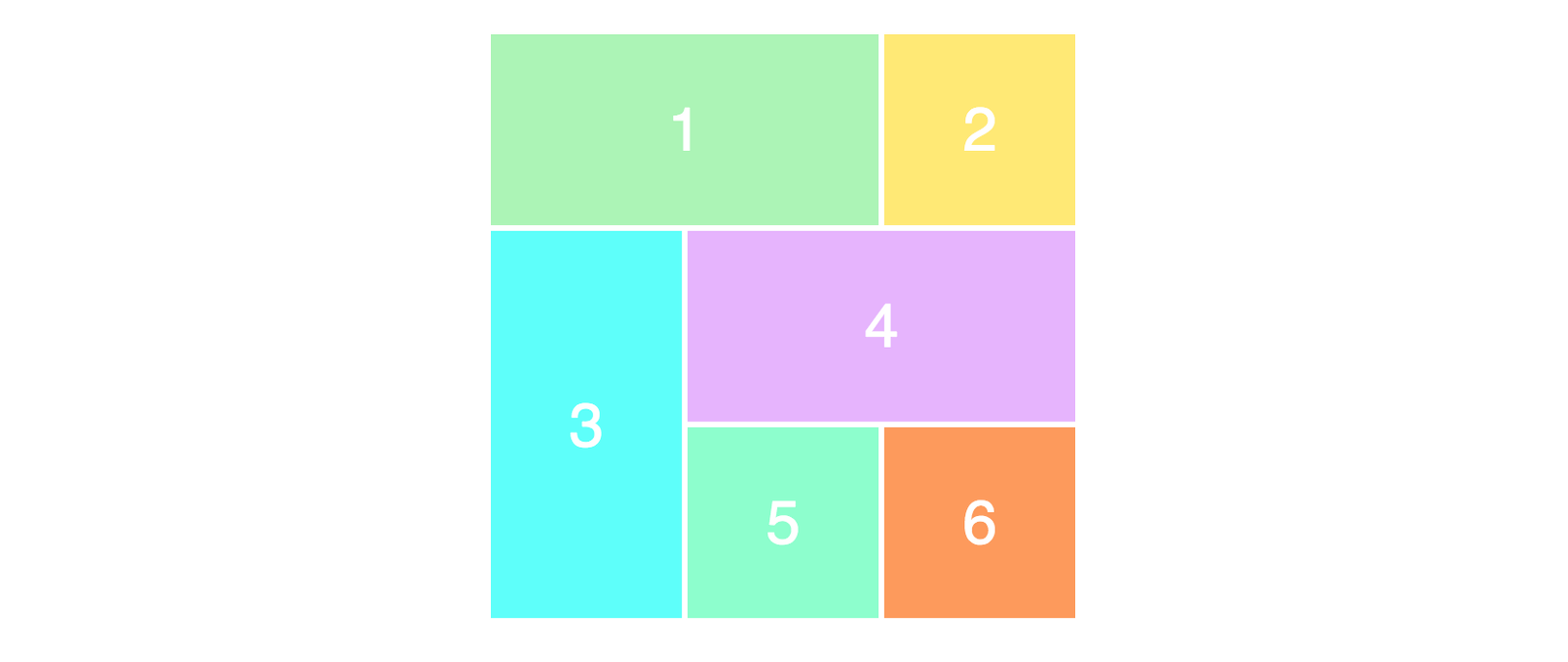
.container {  
 grid-template-columns: 1fr 1fr 1fr; }

.grid\_conteiner\_items:nth-child(1) {  
grid-column-start:3; // ячейка перемещается в 3 колонку и опускаеся во второй ряд  
grid-row-start:2; }

.grid\_conteiner\_items:nth-child(1) {  
grid-column-start:2;  
grid-column-end:5;  
grid-row-start:2; // ячейка начинается со второго ряда второй колонки и завершается на 4 ряду тоесть занимает место 2 рядов  
grid-row-end:4;

.grid\_conteiner\_items:nth-child(1) {  
grid-column:2 / 4;  
grid-row: 2 / 4;

grid-area: 2 / 2 / 4 / 4;  
// grid-area: grid-column-start / grid-row-start / grid-column-end / grid-row-end;}



.item-a { grid-area: header; }  
.item-b { grid-area: main; }  
.item-c { grid-area: sidebar; }  
.item-d { grid-area: footer; }

.container {  
 grid-template-columns: 50px 50px 50px 50px;  
 grid-template-rows: auto;  
 grid-template-areas:   
 "header header header header"  
 "main main . sidebar"  
 "footer footer footer footer";  
}

#### **grid-template-areas**

Определяет шаблон сетки ссылаясь на имена областей, которые заданы с помощью свойства [grid-area](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-area). Повторение названия области приводит к тому, что содержимое охватывает эти ячейки. Точка означает пустую ячейку. Сам синтаксис предоставляет визуализацию структуры сетки.

Значения:

1. **<grid-area-name>** - имя области заданное с помощью [grid-area](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-area);
2. **.** - точка обозначающая пустую ячейку;
3. **none** - области не определены;

.item-a { grid-area: header; }  
.item-b { grid-area: main; }  
.item-c { grid-area: sidebar; }  
.item-d { grid-area: footer; }  
.container { grid-template-columns: 50px 50px 50px 50px;  
grid-template-rows: auto;  
grid-template-areas:   
"header header header header"  
"main main . sidebar"  
"footer footer footer footer"; }

#### **grid-template**

Сокращение для [grid-template-rows](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-template-columns-rows), [grid-template-columns](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-template-columns-rows), и [grid-template-areas](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-template-areas).

Значения:

* **none** - устанавливает все три свойства в их начальное значение;
* **subgrid** - устанавливает [grid-template-rows](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-template-columns-rows) и [grid-template-columns](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-template-columns-rows) в subgrid, и [grid-template-areas](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-template-areas) в его начальное значение;
* **<grid-template-rows> / <grid-template-columns>** - устанавливает [grid-template-columns](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-template-columns-rows) и [grid-template-rows](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-template-columns-rows) в определённое значение, соответственно, и устанавливает [grid-template-areas](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-template-areas) в none;

#### **.container { grid-template: [row1-start] 25px "header header header" [row1-end] [row2-start] "footer footer footer" 25px [row2-end] / auto 50px auto; }**

#### **.container { grid-template-rows: [row1-start] 25px [row1-end row2-start] 25px [row2-end]; grid-template-columns: auto 50px auto; grid-template-areas: "header header header" "footer footer footer"; }**

#### **# grid-column-gap grid-row-gap**

Определяет размер ширины линий. Вы можете думать об этом, как о настройке ширины отступов между столбцами и строками.

Значения:

<line-size> - значение размера;

.container {  
 grid-template-columns: 100px 50px 100px;  
 grid-template-rows: 80px auto 80px;   
 grid-column-gap: 10px;  
 grid-row-gap: 15px; }

#### **grid-gap**

Сокращение для [grid-row-gap](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-column-row-gap) и [grid-column-gap](https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-column-row-gap).

Значение:  
<grid-row-gap> <grid-column-gap> - значения размеров;  
.container{  
grid-template-columns: 100px 50px 100px;  
grid-template-rows: 80px auto 80px;   
grid-gap: 10px 15px; }

<https://tuhub.ru/posts/css-grid-complete-guide#prop-grid-template-areas>