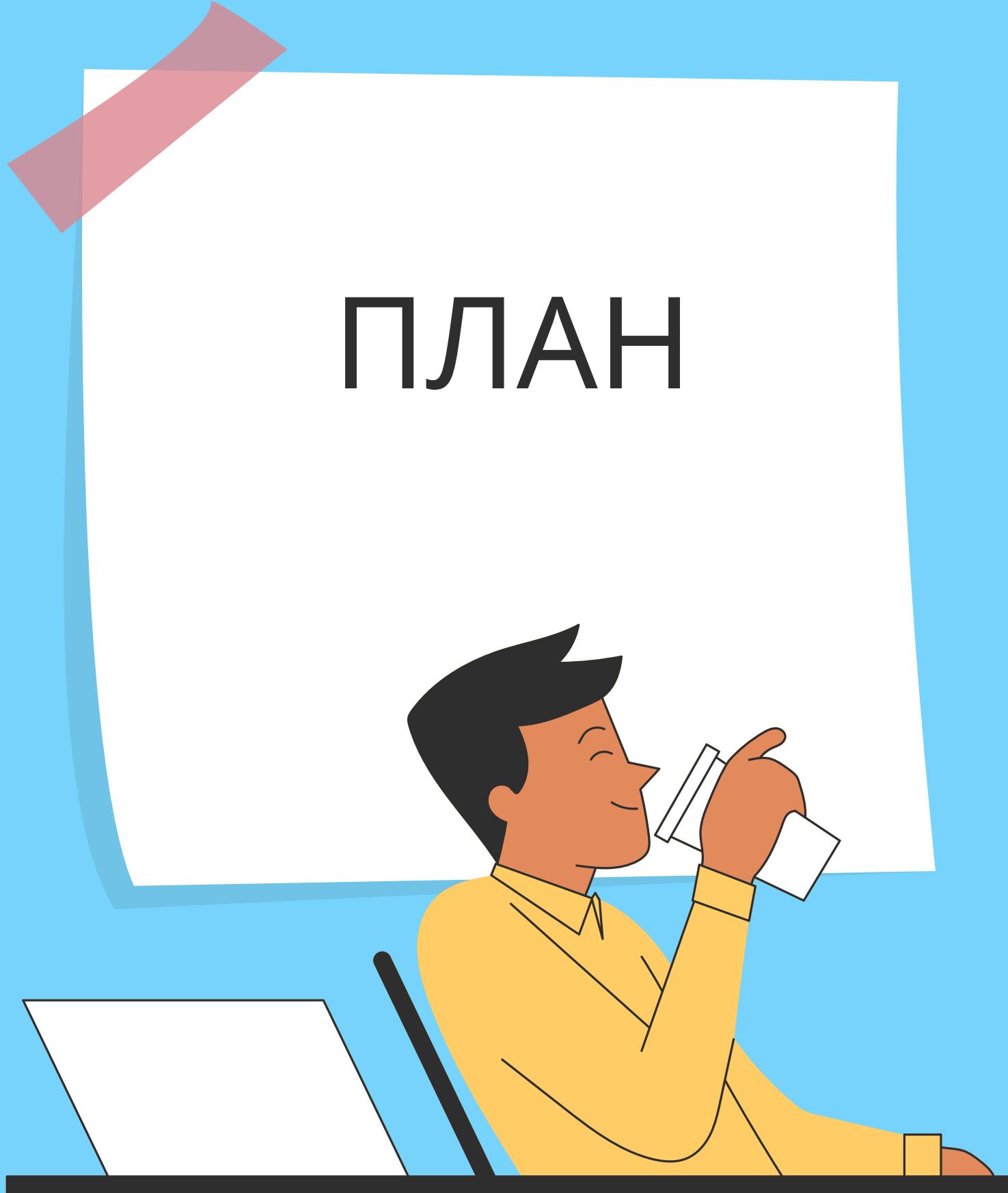




CSS

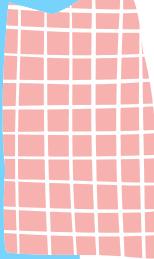
Cascading Style
Sheets
(Каскадные
таблицы стилей.)



1

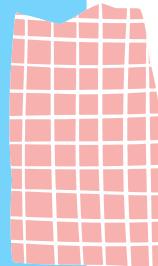
Grid.

GRID



CSS Grid Layout или просто гриды — это удобная технология для раскладки элементов на веб-страницах. В отличие от флексбоксов, одновременно работающих только с одним измерением, гриды дают возможность работать одновременно с двумя: горизонталью и вертикалью.

Грид-контейнер: родительский элемент, к которому применяется свойство `display: grid`.



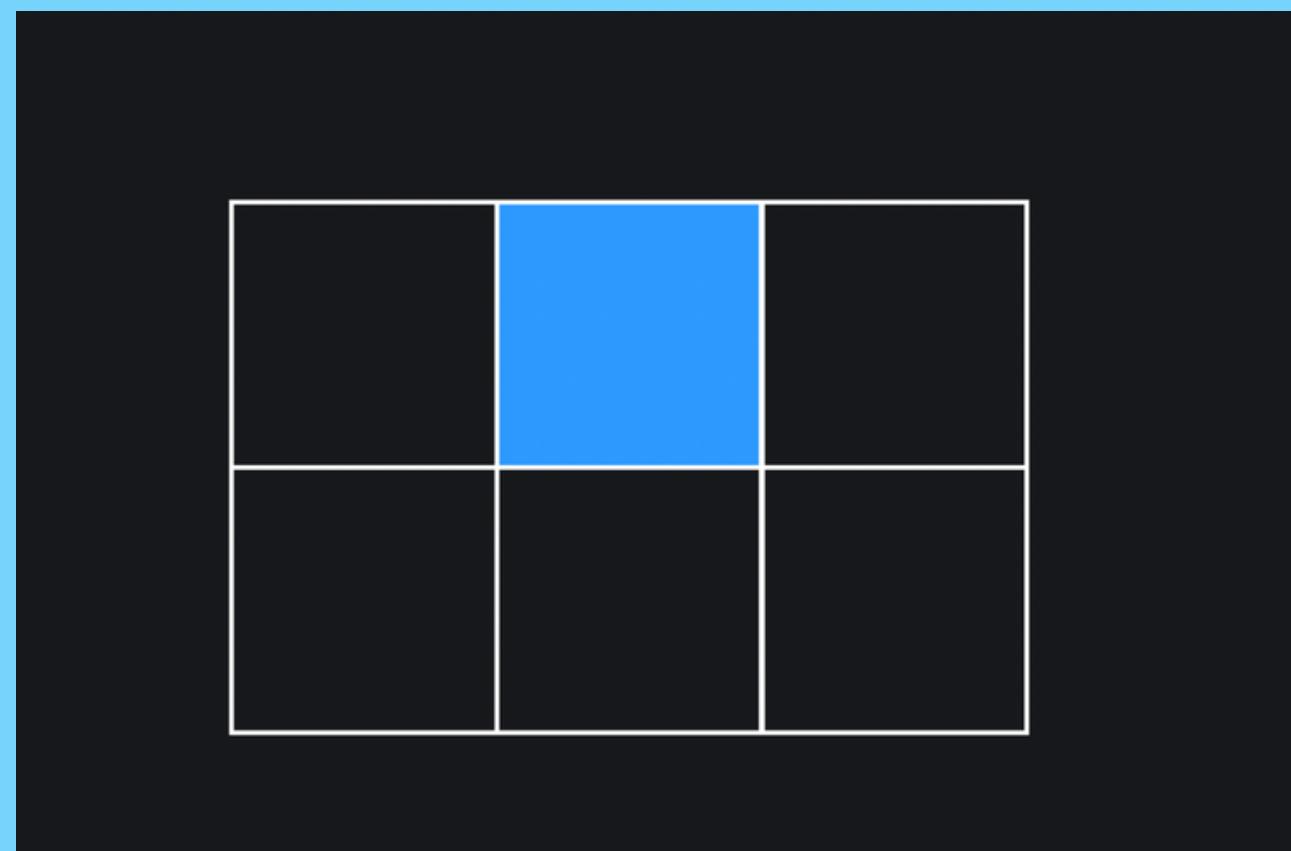
Грид-элемент: дочерний элемент, прямой потомок грид-контейнера.

Подчиняется правилам раскладки гридов.

Грид-линия: разделительная линия, формирующая структуру грида. Может быть как вертикальной (грид-линия колонки), так и горизонтальной (грид-линия ряда). Располагается по обе стороны от колонки или ряда.

Используется для привязки грид-элементов.

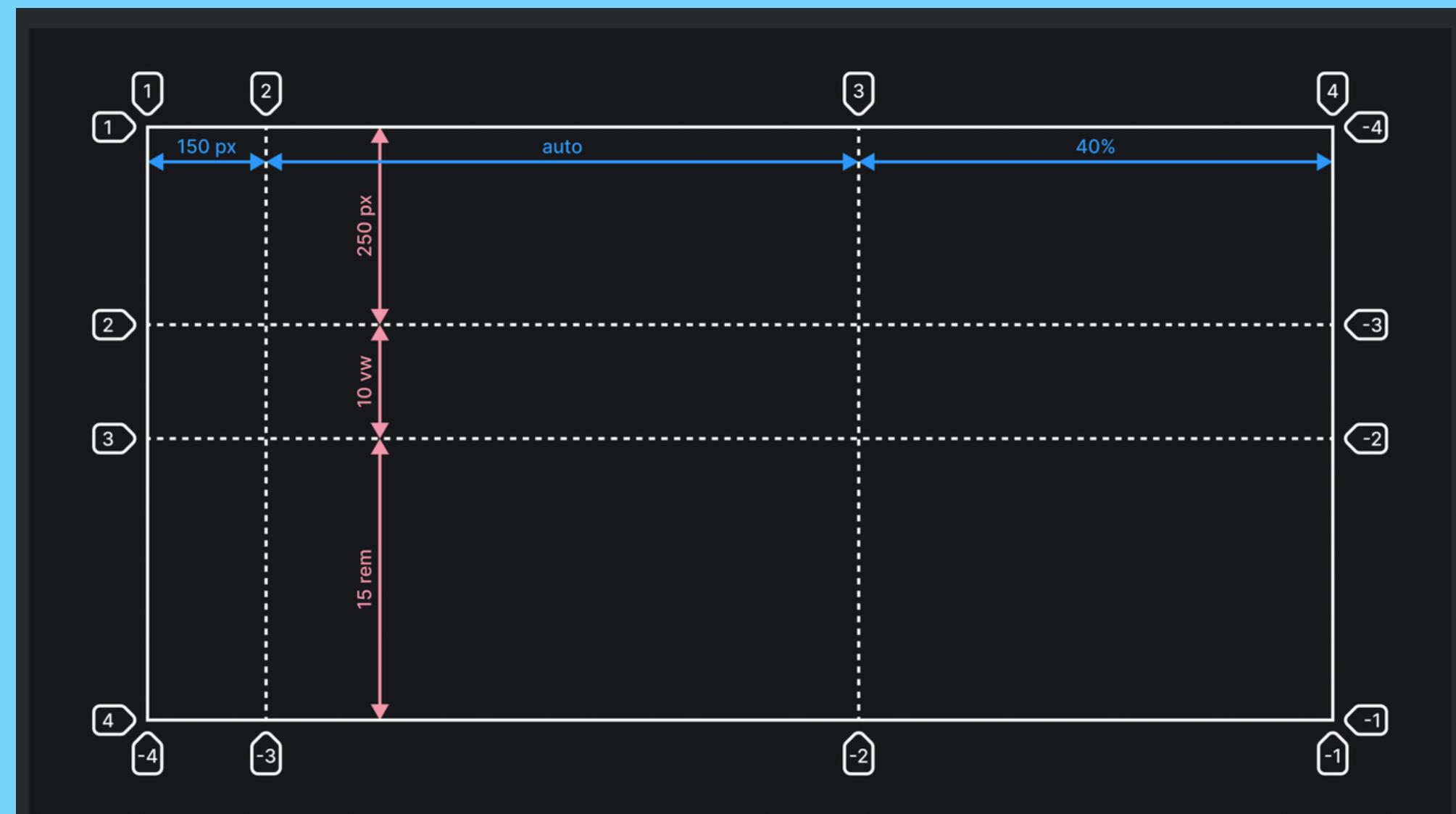
Грид-ячейка: пространство между соседними грид-линиями. Единица грид-сетки.



display: grid: Основное свойство, которое применяется к родительскому элементу, чтобы создать сетку. Устанавливает элемент-контейнер в режим сетки.

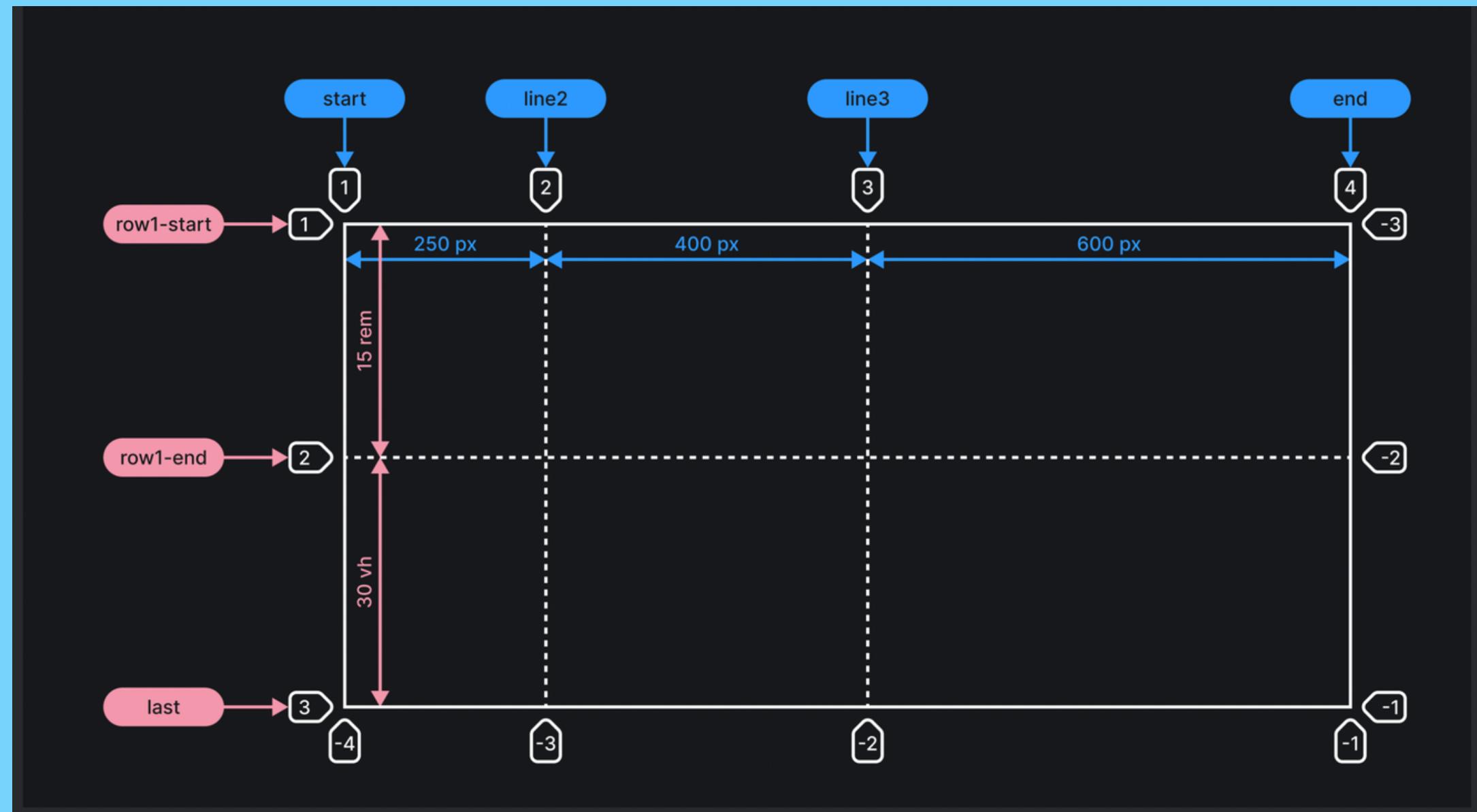
grid-template-columns и **grid-template-rows**: Эти свойства определяют структуру сетки, задавая количество и размеры столбцов и строк, соответственно

```
1 .container {  
2   display: grid;  
3   /* Будет создано 3 колонки */  
4   grid-template-columns: 150px auto 40%;  
5   /* Будет создано 3 ряда */  
6   grid-template-rows: 250px 10vw 15rem;  
7 }
```



Можно явно именовать грид-линии, используя для этого квадратные скобки:

```
css
1 .container {
2   display: grid;
3   grid-template-columns: [start] 250px [line2] 400px [line3] 600px [end];
4   grid-template-rows: [row1-start] 15rem [row1-end] 30vh [last];
5 }
```



Если нужны одинаковые колонки или ряды, то можно воспользоваться функцией `repeat()`.

Будет создано 3 колонки по 250 пикселей:

css

```
1 .container {  
2   display: grid;  
3   grid-template-columns: repeat(3, 250px);  
4 }
```

С появлением гридов у нас появилась и новая единица измерения: fr
fr (от fraction — доля, часть) отвечает за свободное пространство внутри
грид-контейнера.

Например, этот код создаст три колонки, каждая из которых будет
занимать 1/3 ширины родителя:

```
1 .container {  
2   display: grid;  
3   grid-template-columns: repeat(3, 1fr);  
4 }
```

Что аналогично записи:

css

```
1 .container {  
2   display: grid;  
3   grid-template-columns: 1fr 1fr 1fr;  
4 }
```

Свободное пространство рассчитывается после того, как место отдано всем фиксированным размерам. К примеру, в коде ниже сначала будет создана колонка шириной 200 пикселей, а затем свободное пространство — ширина родителя минус 200 пикселей — будет поделено между остальными колонками. Каждая будет занимать ширину $(100\% - 200\text{px}) / 2$:

css

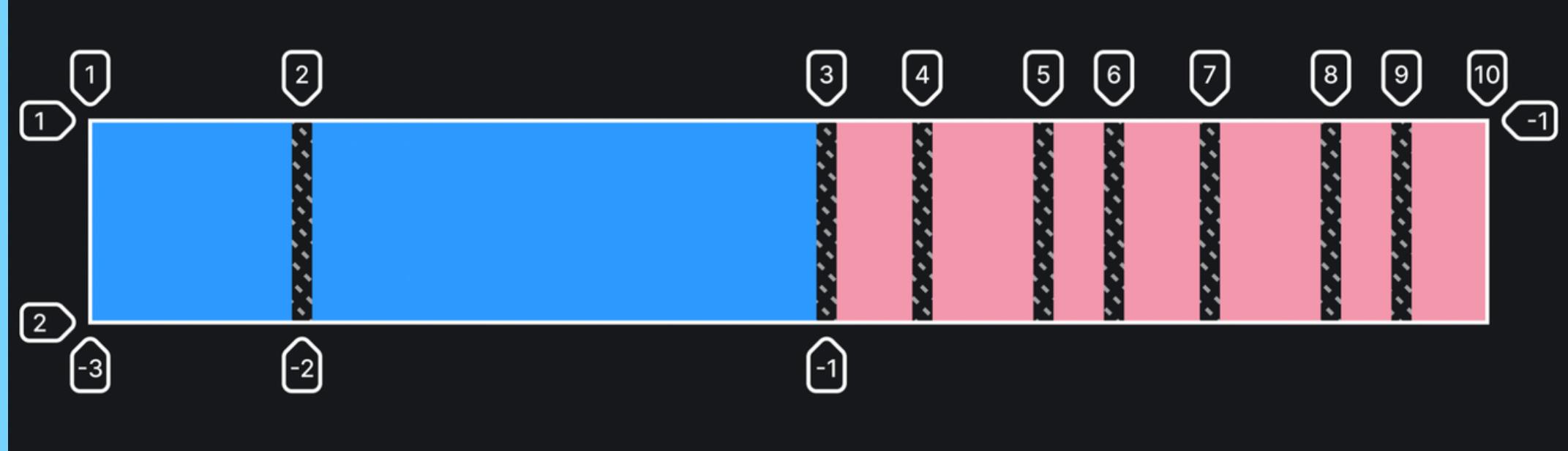
```
1 .container {  
2   display: grid;  
3   grid-template-columns: 1fr 200px 1fr;  
4 }
```

grid-auto-columns, grid-auto-rows

Если элементов внутри грид-контейнера больше, чем может поместиться в заданные явно ряды и колонки, то для них создаются автоматические, неявные ряды и колонки. При помощи свойств `grid-auto-columns` и `grid-auto-rows` можно управлять размерами этих автоматических рядов и колонок.

```
1 .container {  
2   display: grid;  
3   grid-template-rows: 50px 140px;  
4   grid-auto-rows: 40px;  
5   gap: 20px;  
6 }
```

```
1 .container {  
2   display: grid;  
3   grid-template-columns: 200px 500px;  
4   grid-auto-columns: 75px 100px 50px;  
5   grid-auto-flow: column;  
6   gap: 20px;  
7 }
```



Важно указать при помощи `grid-auto-flow: column`, что элементы должны вставлять в колонки, чтобы элементы без контента были видны. Как видите, автоматически создаются колонки размером 75, 100 и затем 50 пикселей. И так до тех пор, пока элементы не закончатся.

grid-auto-flow

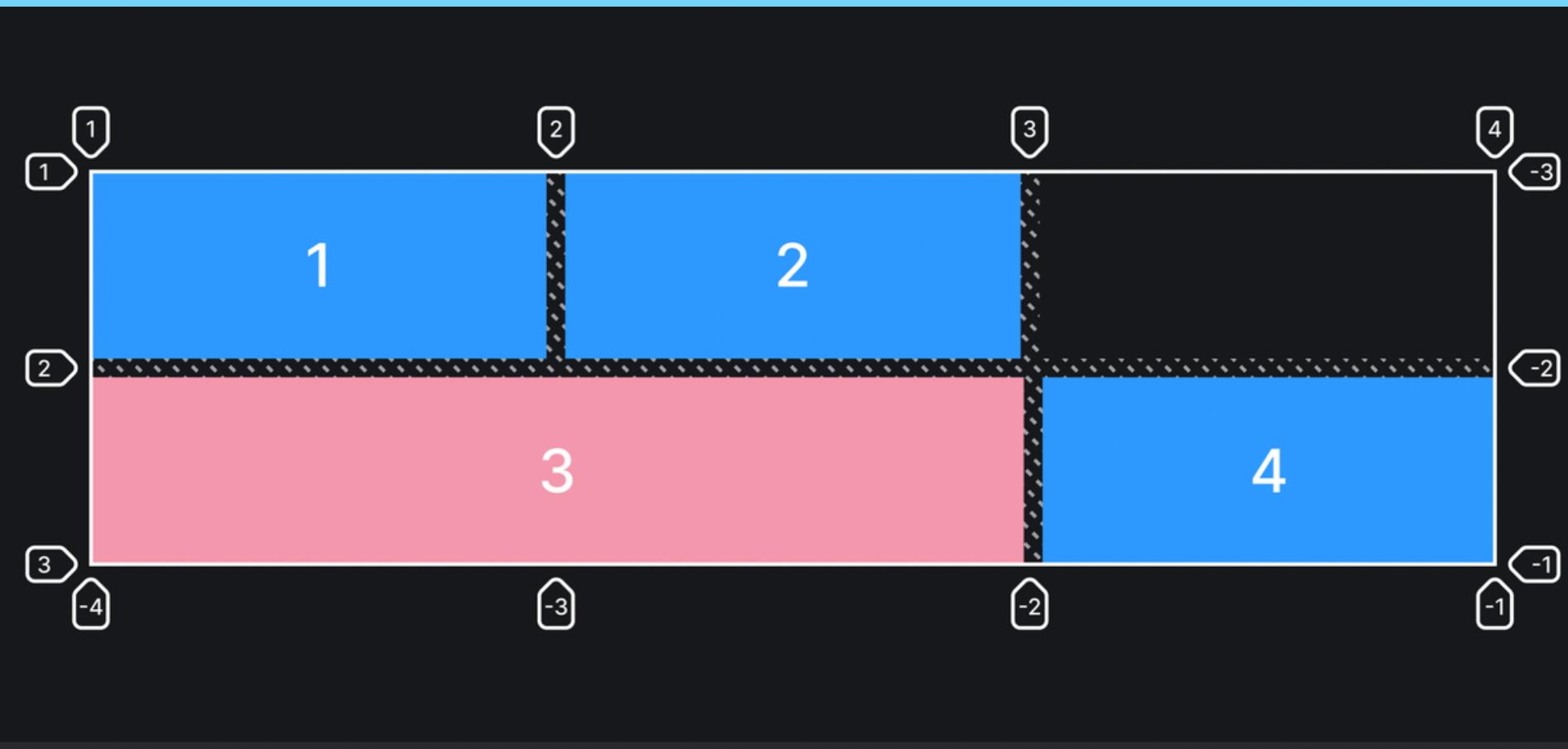
Если грид-элементов больше, чем явно объявленных колонок или рядов, то они автоматически размещаются внутри родителя. А вот каким образом — в ряд или в колонку — можно указать при помощи свойства `grid-auto-flow`. По умолчанию значение у этого свойства `row`, лишние элементы будут выстраиваться в ряды в рамках явно заданных колонок.

Возможные значения:

- `row` (значение по умолчанию) — автоматически размещаемые элементы выстраиваются в ряды.
- `column` — автоматически размещаемые элементы выстраиваются в колонки.
- `dense` — браузер старается заполнить дырки (пустые ячейки) в разметке, если размеры элементов позволяют. Можно сочетать с остальными значениями.

Принципы работы этого свойства удобнее всего изучать на примере, когда есть большой блок, который не помещается в одну грид-ячейку.

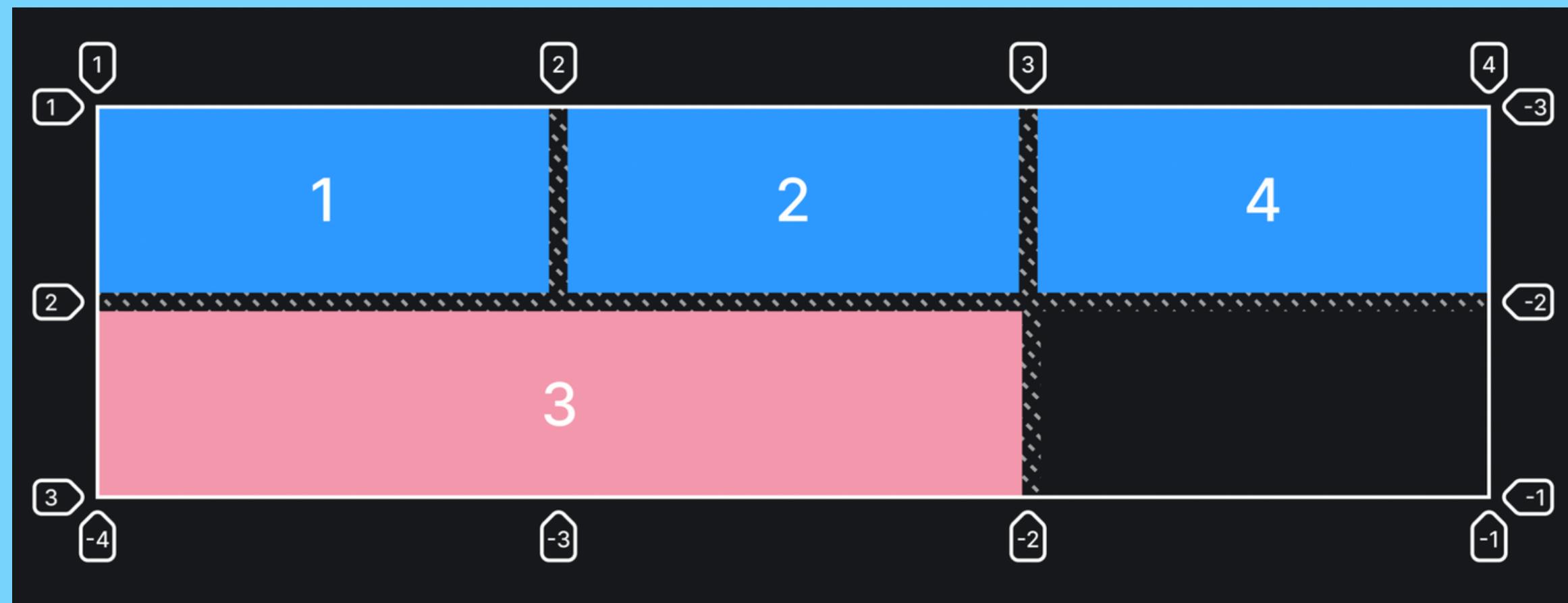
```
1 .container {  
2   display: grid;  
3   /* Три колонки */  
4   grid-template-columns: auto auto auto;  
5   /* Два ряда */  
6   grid-template-rows: auto auto;  
7   /* Автоматическое размещение в ряд */  
8   grid-auto-flow: row;  
9   /* Отступы между ячейками */  
10  gap: 20px;  
11 }  
12  
13 .item3 {  
14   /* Занимает один ряд и  
15   растягивается на две колонки */  
16   grid-column: span 2;  
17 }
```



Как видите, третий элемент не поместился в последнюю ячейку первого ряда и был перенесён в следующий ряд. Следующий за ним четвёртый элемент встал в ближайшую доступную ячейку во втором ряду.

Добавим к значению свойства grid-auto-flow слово dense:

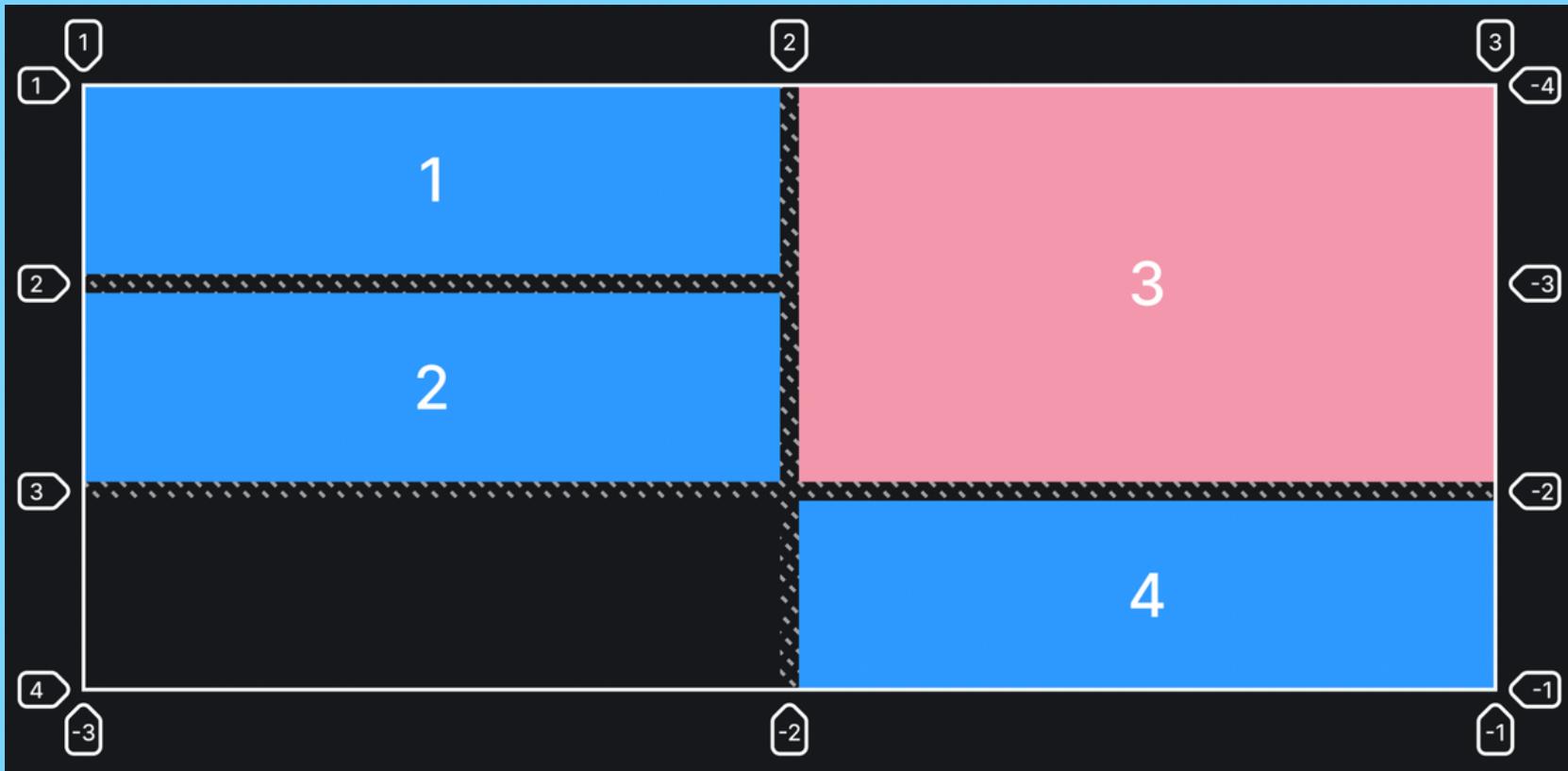
```
1 .container {  
2     /* Автоматическое размещение в ряд */  
3     grid-auto-flow: row dense;  
4 }
```



Теперь четвёртый элемент встал в ряд, но занял при этом пустую ячейку в первом ряду. Браузер старается закрыть все дырки в сетке, переставляя подходящие элементы на свободные места.

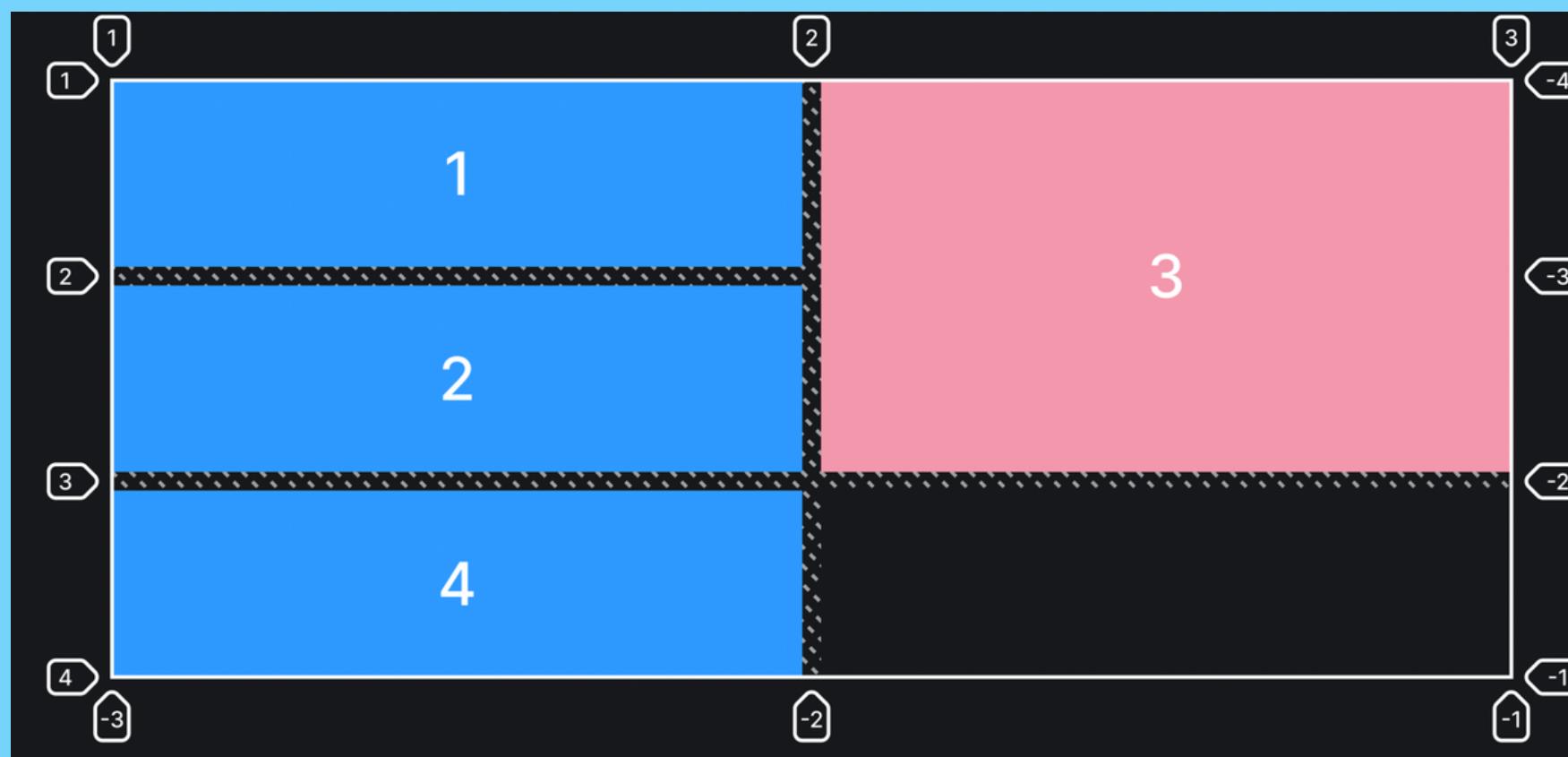
Посмотрим пример со значением
column:

```
1 .container {  
2   grid-template-columns: auto auto;  
3   grid-template-rows: auto auto auto;  
4   /* Автоматическое размещение в колонку */  
5   grid-auto-flow: column;  
6 }  
7  
8 .item3 {  
9   grid-row: span 2;  
10 }
```



Видим аналогичную картину: третий элемент не поместился в последнюю ячейку первой колонки и встал во вторую колонку. Следующий за ним четвёртый элемент встал ниже во второй колонке. Добавим `dense` к текущему значению:

```
1 .container {  
2   /* Автоматическое размещение в ряд */  
3   grid-auto-flow: column dense;  
4 }
```



В результате четвёртый элемент занял пустую ячейку в первой колонке.

grid-template-areas

Позволяет задать шаблон сетки расположения элементов внутри грид-контейнера. Имена областей задаются при помощи свойства `grid-area`. Текущее свойство `grid-template-areas` просто указывает, где должны располагаться эти грид-области.

Возможные значения:

- `none` (значение по умолчанию) — области сетки не задано имя.
- `.` — означает пустую ячейку.
- название — название области, может быть абсолютно любым словом или даже эмодзи.

Обратите внимание, что нужно называть каждую из ячеек.

Например, если шапка или подвал нашего сайта будут занимать все три существующие колонки, то нужно будет трижды написать названия этих областей.

```
.container {
    display: grid;
    grid-template-columns: repeat(3, 500px);
    grid-template-rows: repeat(4, 1fr);
    grid-template-areas:
        "header header header"
        "content content ."
        "content content ."
        "footer footer footer";
}

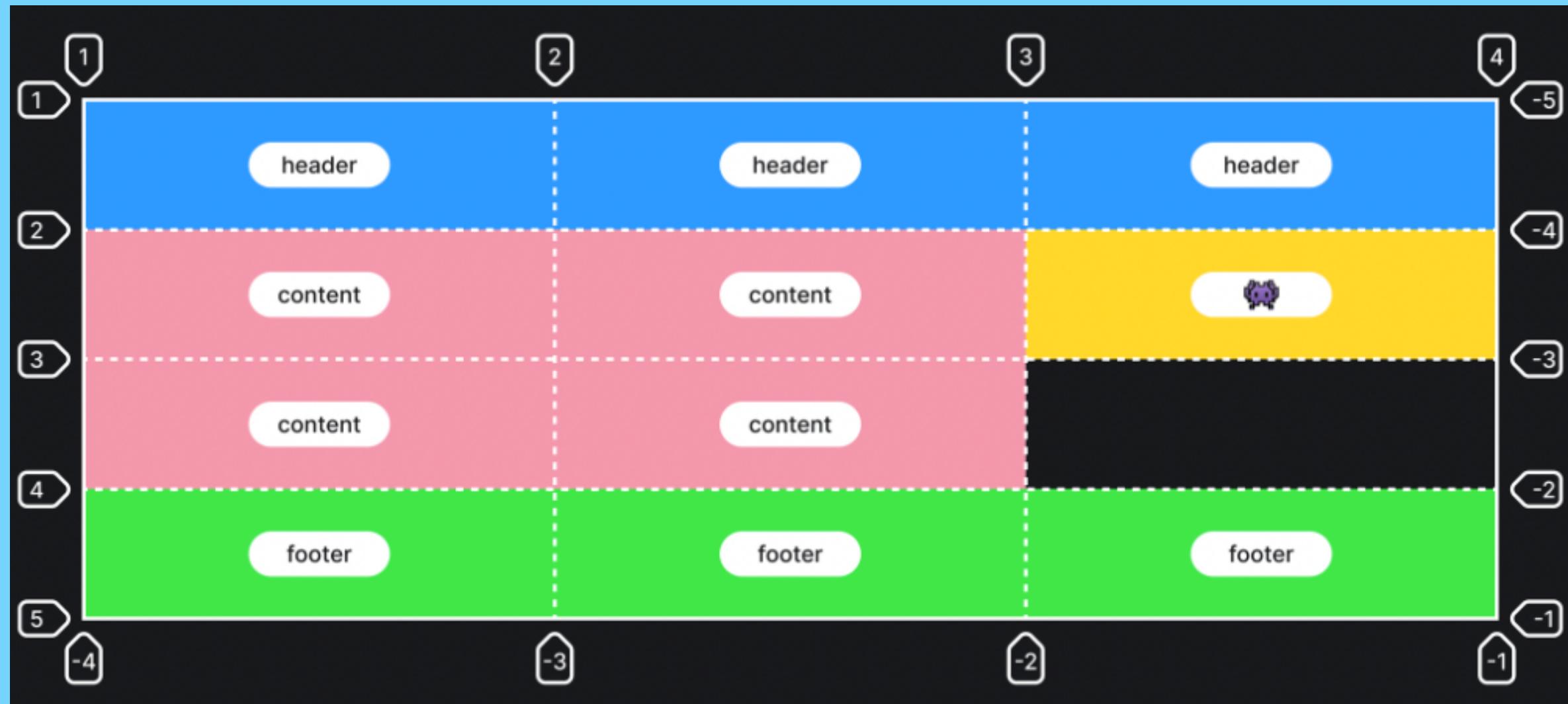
.item1 {
    grid-area: header;
}

.item2 {
    grid-area: content;
}

.item3 {
    grid-area: .;
}

.item4 {
    grid-area: footer;
}
```

```
.container {  
  display: grid;  
  grid-template-columns: repeat(3, 500px);  
  grid-template-rows: repeat(4, 1fr);  
  grid-template-areas:  
    "header header header"  
    "content content ."  
    "content content ."  
    "footer footer footer";  
}  
  
.item1 {  
  grid-area: header;  
}  
  
.item2 {  
  grid-area: content;  
}  
  
.item3 {  
  grid-area: .;  
}  
  
.item4 {  
  grid-area: footer;  
}
```

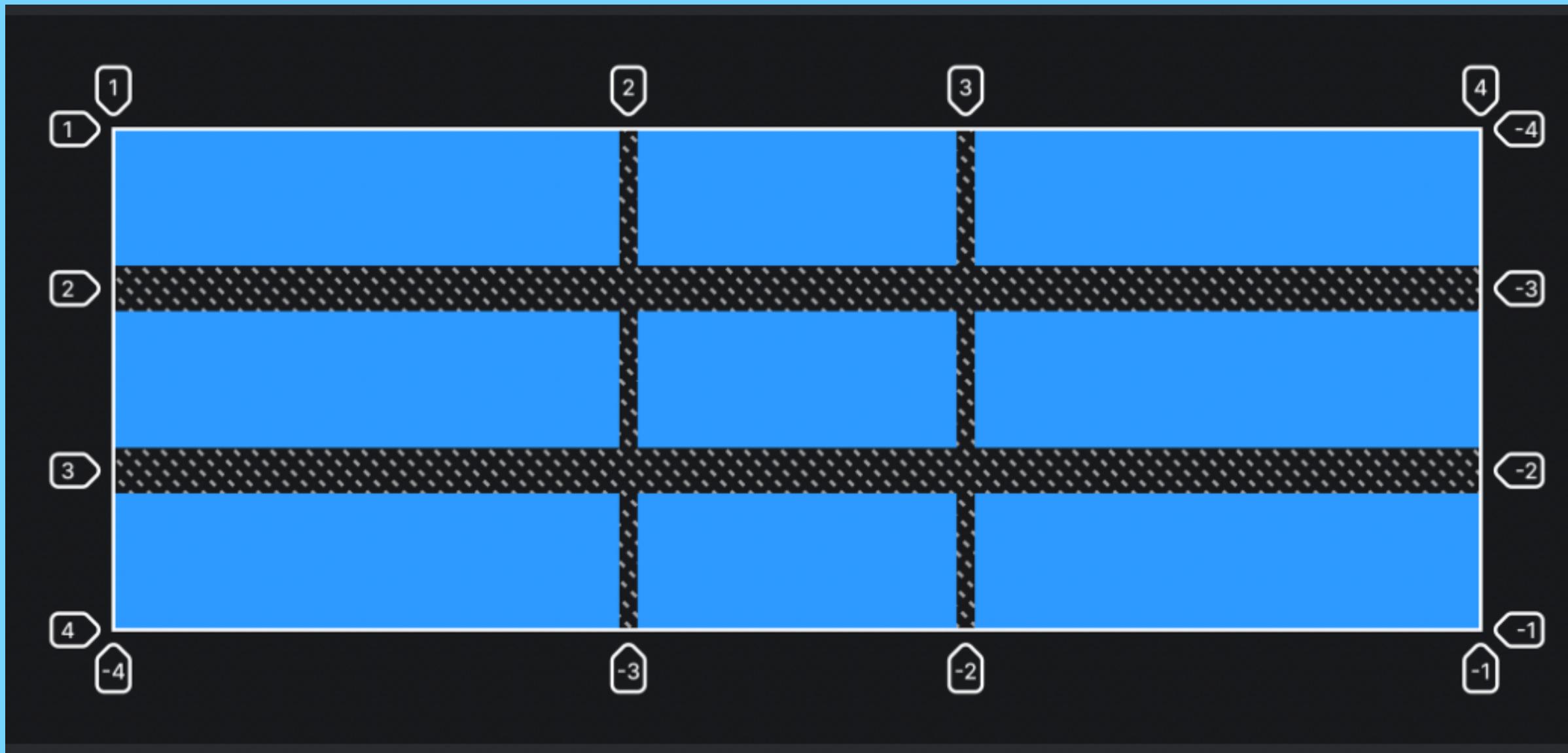


Обратите внимание, что между строками не ставятся запятые или какие-то другие символы, имена разделяются пробелами.

row-gap, column-gap

Задают отступы между рядами или колонками.

```
1 .container {  
2   display: grid;  
3   grid-template-columns: 1fr 350px 1fr;  
4   grid-template-rows: repeat(3, 150px);  
5   /* Отступы между рядами */  
6   row-gap: 50px;  
7   /* Отступы между колонками */  
8   column-gap: 20px;  
9 }
```



gap

Шорткат для записи значений свойств row-gap и column-gap.
Значения разделяются пробелом:

css

```
1 .container {
2   display: grid;
3   grid-template-columns: 1fr 350px 1fr;
4   grid-template-rows: repeat(3, 150px);
5   gap: 50px 20px;
6 }
```

justify-content

Свойство, с помощью которого можно выровнять элементы вдоль оси строки. Данное свойство работает, только если общая ширина столбцов меньше ширины контейнера сетки. Другими словами, вам нужно свободное пространство вдоль оси строки контейнера, чтобы выровнять его столбцы слева или справа.

start — выравнивает сетку по левой стороне грид-контейнера.

end — выравнивает сетку по правой стороне грид-контейнера.

center — выравнивает сетку по центру грид-контейнера.

stretch — масштабирует элементы, чтобы сетка могла заполнить всю ширину грид-контейнера.

space-around — одинаковое пространство между элементами и полуразмерные отступы по краям.

space-evenly — одинаковое пространство между элементами и полноразмерные отступы по краям.

space-between — одинаковое пространство между элементами без отступов по краям.

justify-items

Свойство, с помощью которого задаётся выравнивание грид-элементов по горизонтальной оси. Применяется ко всем элементам внутри грид-родителя.

start — выравнивает элемент по начальной линии.

end — выравнивает элемент по конечной линии.

center — выравнивает элемент по центру грид-ячейки.

stretch — растягивает элемент на всю ширину грид-ячейки.

align-items

Свойство, с помощью которого можно выровнять элементы по вертикальной оси внутри грид-контейнера.

start — выравнивает элемент по начальной (верхней) линии.

end — выравнивает элемент по конечной (нижней) линии.

center — выравнивает элемент по центру грид-ячейки.

stretch — растягивает элемент на всю высоту грид-ячейки.

Свойства грид-элементов

grid-column-start, grid-column-end
grid-row-start, grid-row-end

Определяют положение элемента внутри грид-сетки при помощи указания на конкретные направляющие линии.

- название или номер линии — может быть порядковым номером или названием конкретной линии.
- span число — элемент растягивается на указанное количество ячеек.
- span имя — элемент будет растягиваться до следующей указанной линии.
- auto — означает автоматическое размещение, автоматический диапазон клеток или дефолтное растягивание элемента, равное одному.

```
7 .item1 {  
8   grid-column-start: 2;  
9   grid-column-end: five;  
10  grid-row-start: row1-start;  
11  grid-row-end: 3;  
12 }
```

[Скопировать](#)

grid-column, grid-row

Свойства-шорткаты для grid-column-start, grid-column-end и grid-row-start, grid-row-end соответственно.

Значения для *-start и *-end разделяются слэшем.

Можно использовать ключевое слово `span`, буквально говорящее «растянись на столько-то». А на сколько, указывает стоящая за ним цифра.

css

```
1 .item1 {  
2   grid-column: 3 / span 2;  
3   grid-row: 3 / 4;  
4 }
```

 [Скопировать](#)

grid-area

Свойство grid-area даёт название элементу чтобы можно было ссылаться на него с помощью шаблона созданного через grid-template-areas свойство.

Свойство grid-area является сокращенным свойством для grid-row-start, grid-column-start, grid-row-end и grid-column-end, определяя размер и расположение элемента сетки.

```
1 .item1 {  
2   /* Займёт область content внутри грид-сетки */  
3   grid-area: content;  
4 }
```

```
1 .item2 {  
2   grid-area: 1 / col4-start / last-line / 6;  
3 }
```

justify-self

С помощью этого свойства можно установить горизонтальное выравнивание для отдельного элемента, отличное от выравнивания, заданного грид-родителю.

Возможные значения аналогичны значениям свойства justify-items.

align-self

А это свойство, как нетрудно догадаться, выравнивает отдельный элемент по вертикальной оси. Возможные значения аналогичны значениям свойства align-items.

place-self

Шорткат для одновременного указания значений свойствам justify-self и align-self.

Возможные значения:

- auto (значение по умолчанию) — стандартное значение, можно использовать для сброса ранее заданных значений.
- align-self justify-self — первое значение задаёт значение свойству align-self, второе значение устанавливает значение свойства justify-self. Если указано всего одно значение, то оно устанавливается для обоих свойств. Например, place-self: center отцентрирует элемент по горизонтальной и по вертикальной осям одновременно.



**THANK
YOU!**

6

Have a
great day
ahead.