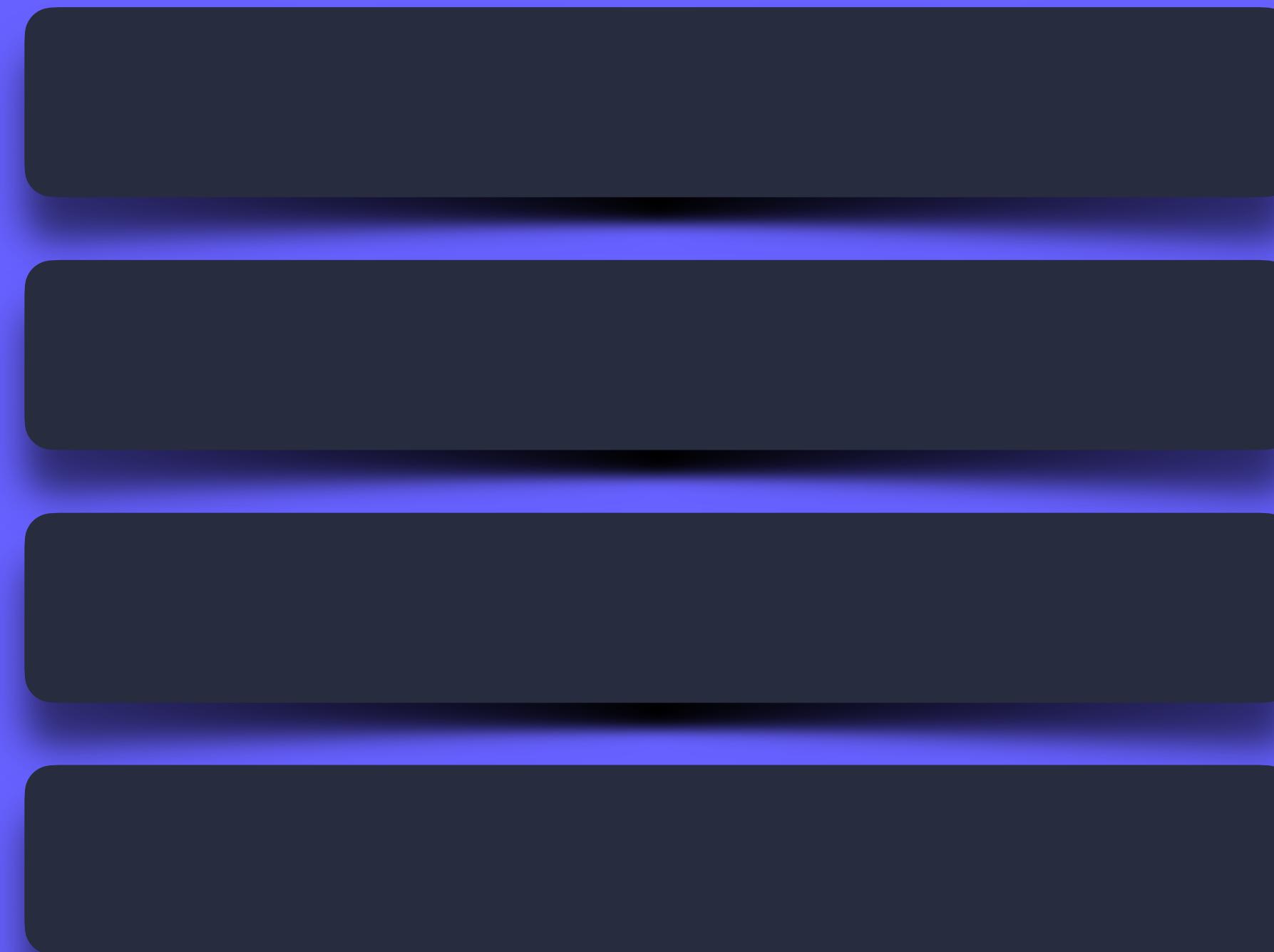


Grid

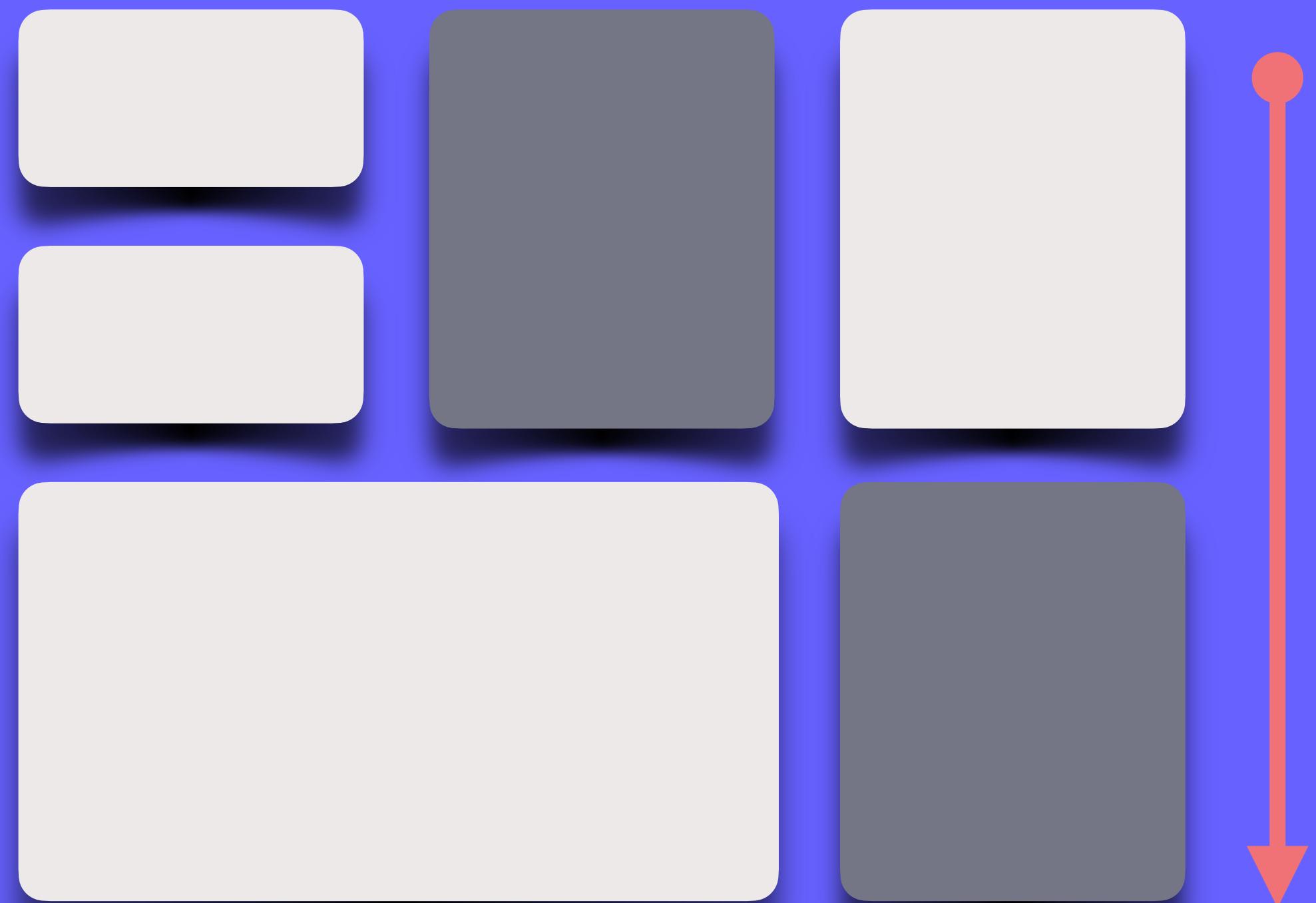
Flexbox vs Grid

Flexbox одновременно может работать только со строкой или столбцом, но не с обоими сразу.



Flexbox эффективно справляется с выравниванием, распределением и направлением элементов страницы.

Grid (сетка) представляет собой мощный двухмерный макет, при использовании которого можно одновременно работать со строками и столбцами.

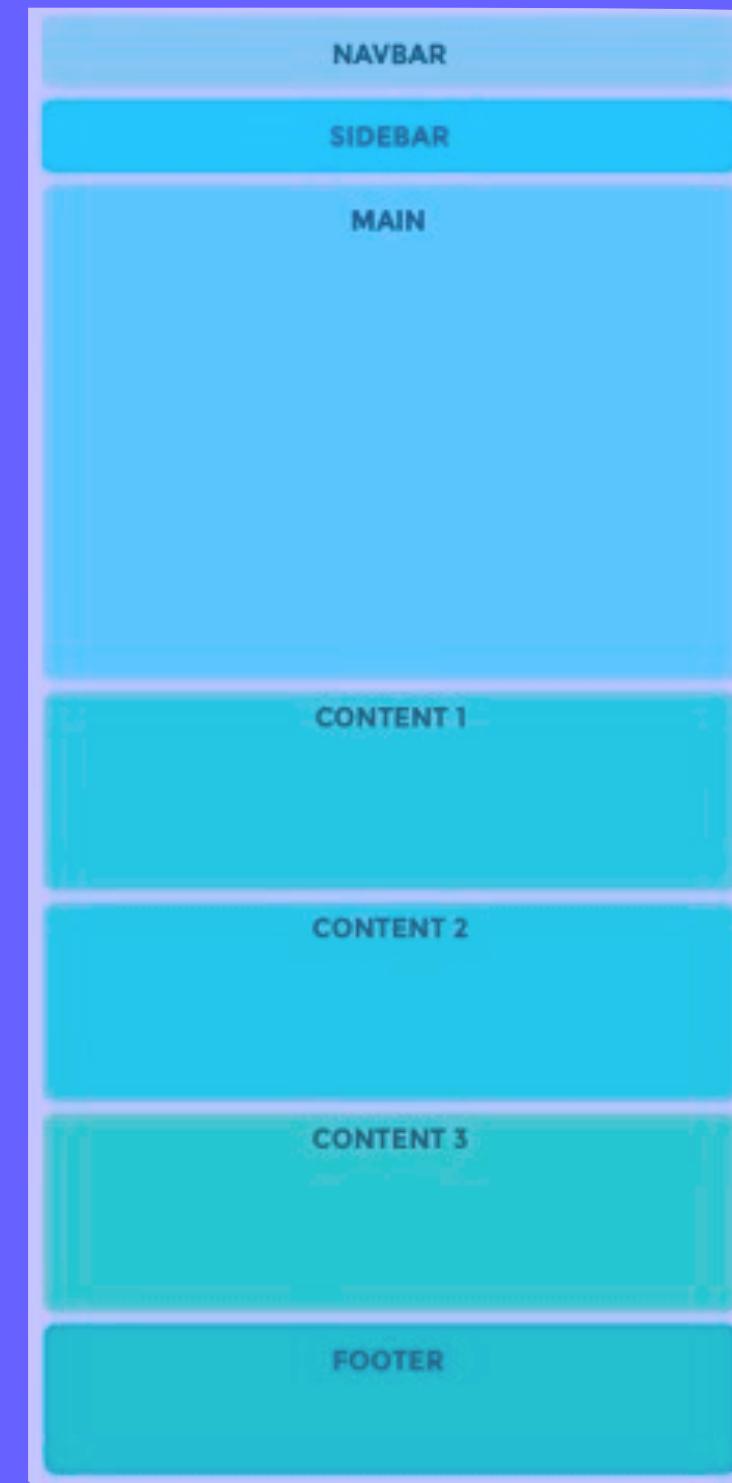
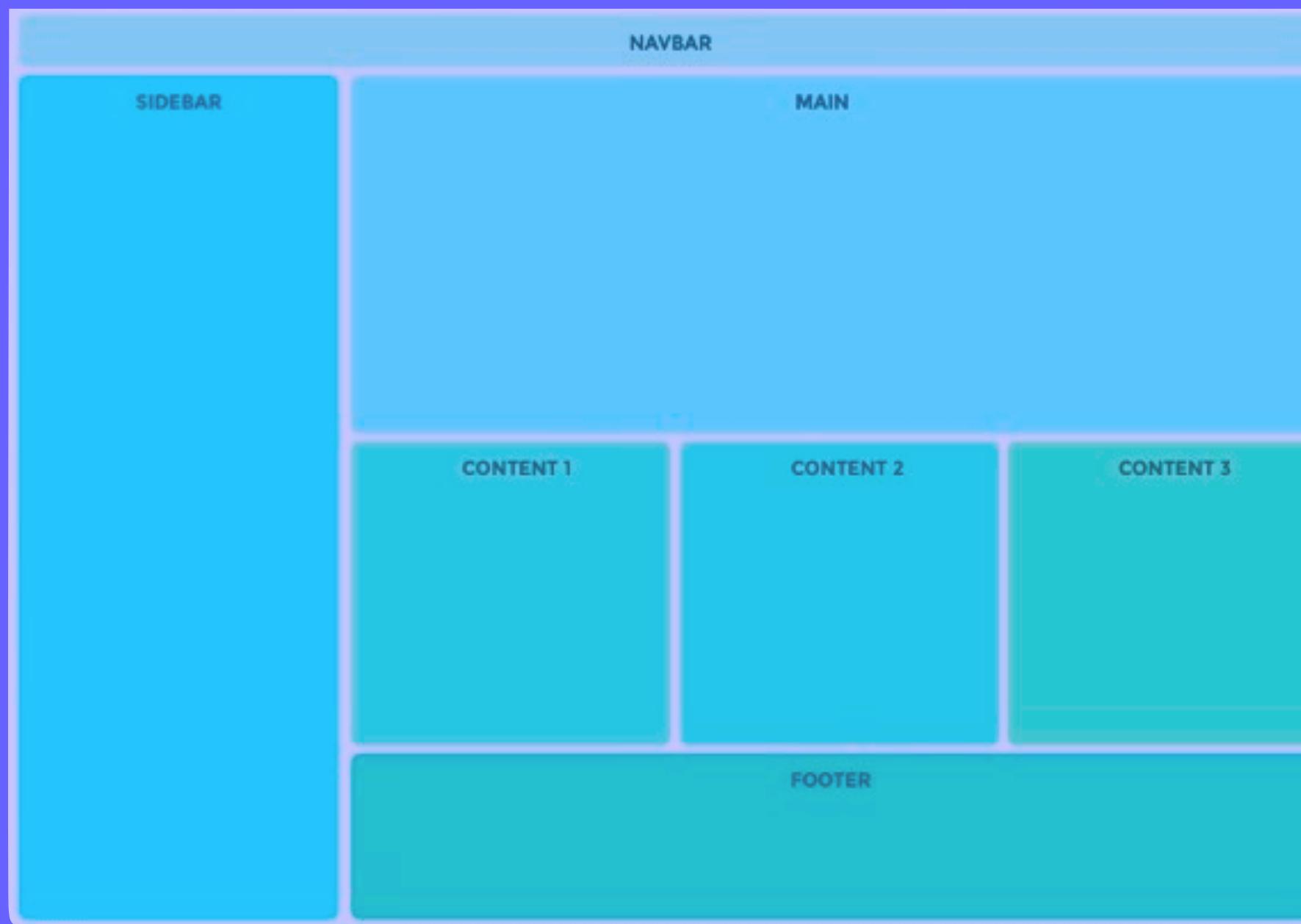


Преимущество этой технологии перед Flexbox и другими видами макетов в ее возможности работать двухмерно.

Где применять Grid?

RESPONSIVE CSS GRID

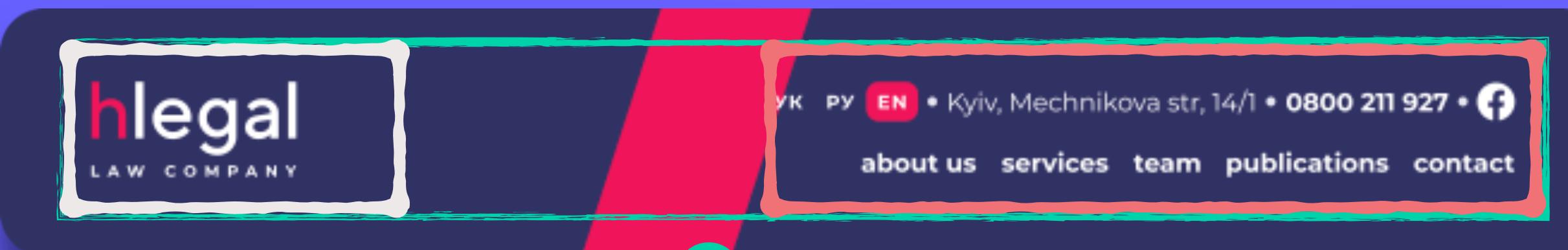
ONLY HTML & CSS



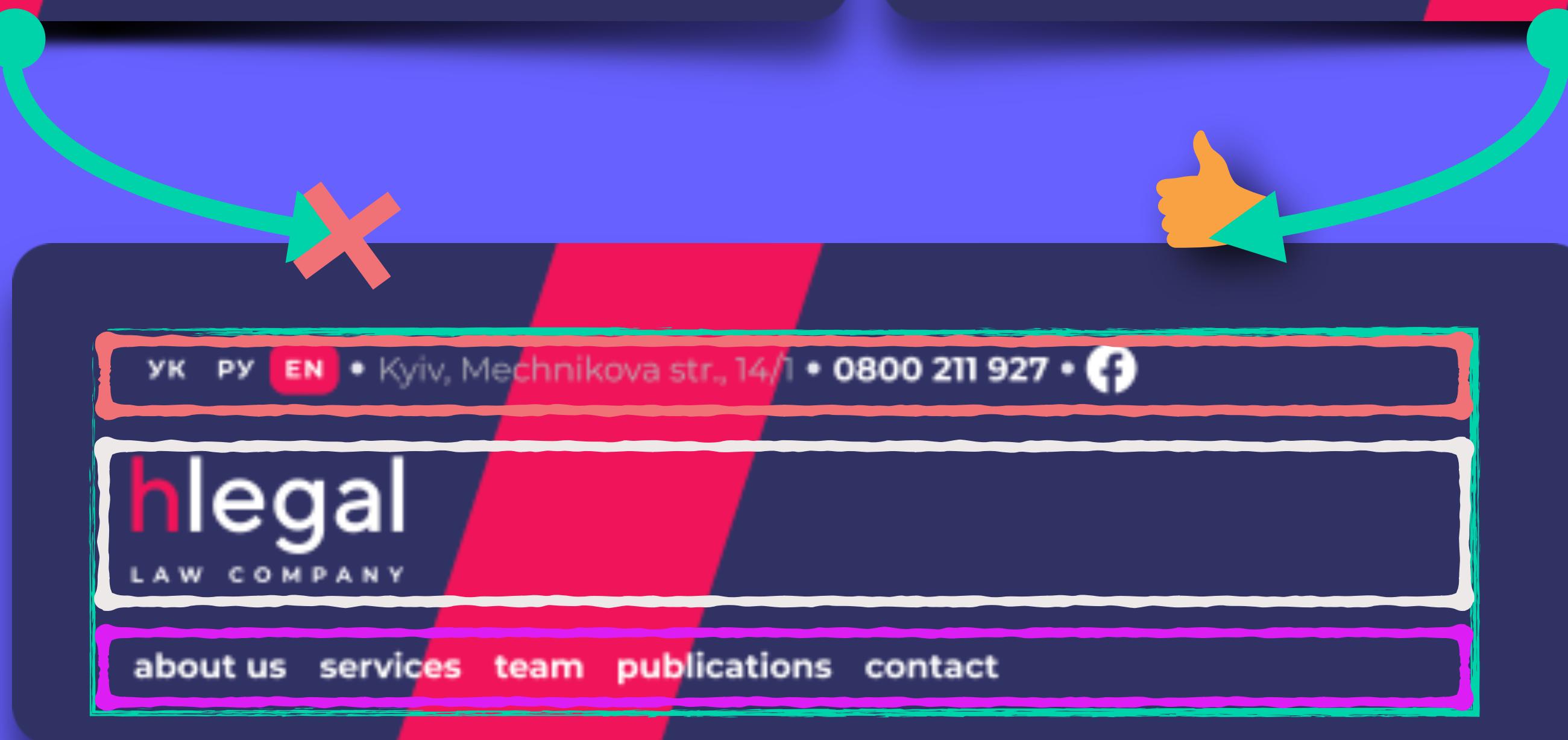
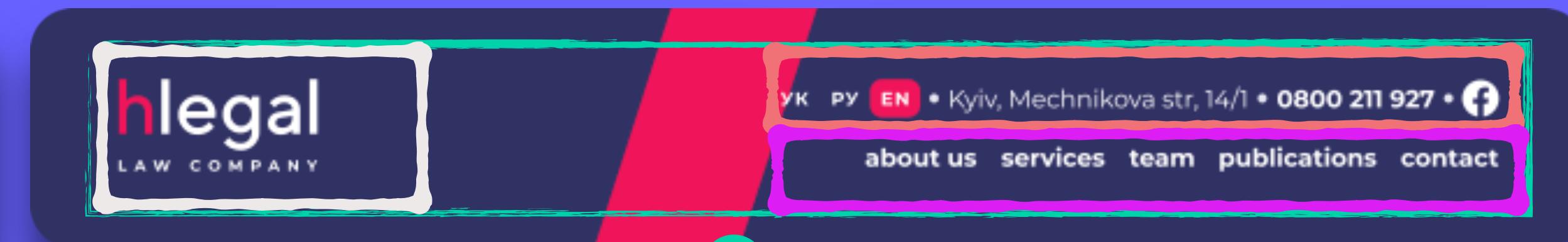
WEB TO
MOBILE
DESIGN

Где применять Grid?

Flexbox



Grid



Создание шаблона

index.html

```
<div class="page-wrap">
  <div class="grid-wrap">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
    <div class="grid-item">7</div>
    <div class="grid-item">8</div>
  </div>
</div>
```

index.scss

```
body {
  margin: 0;
}

* {
  box-sizing: border-box;
}

.page-wrap {
  min-height: 100vh;
  background-color: rgb(22, 117, 206);
}

.grid-item {
  border: 1px solid rgb(88, 220, 238);
  color: rgb(247, 97, 117);
  font-size: 45px;
  text-align: center;
  padding: 30px;
}
```

результат

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

Grid template columns

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: 200px 200px;  
}
```

Для того чтобы создать grid контейнер, мы объявляем на элементе `display:grid` или `inline-grid`. После этого все прямые дети этого элемента станут элементами сетки.

`Grid-template-columns` управляет шириной и количеством колонок

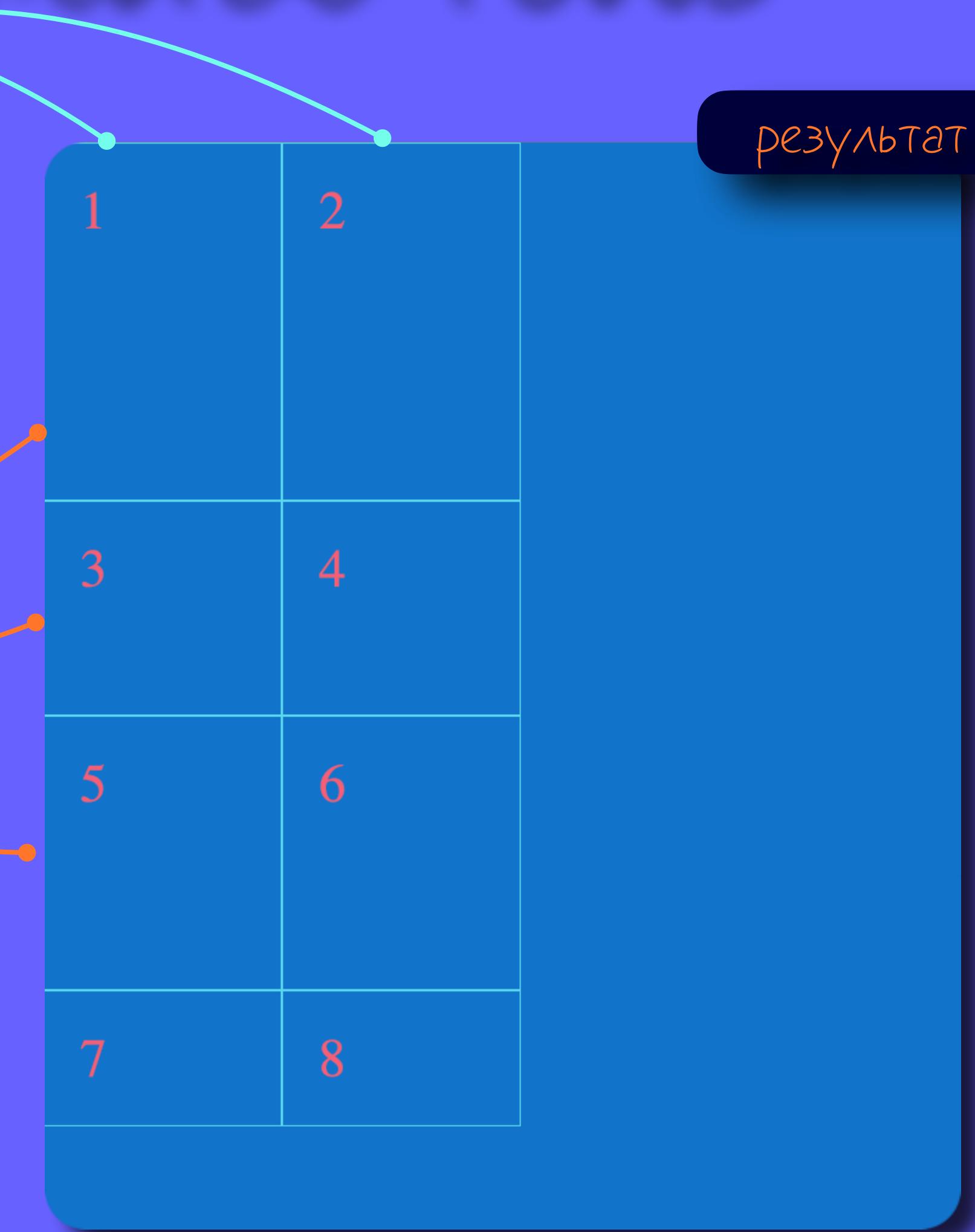
1	2
3	4
5	6
7	8

результат

Grid template rows

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: 200px 200px;  
  grid-template-rows: 300px 180px 230px;  
}
```



Grid-template-rows управляет высотой ряда. Если высота для ряда не проставлена она будет высчитываться относительно содержимого

% and repeat

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: 25% 25% 25% 25%;  
  grid-template-rows: 300px 180px 230px;  
}
```



результат

1	2	3	4
5	6	7	8

Проставлять для каждой колонки (ряда) одинаковую ширину (высоту) достаточно не
практичный метод, поэтому на помощь приходит функция `repeat(count, value)`

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: repeat(4, 25%);  
  grid-template-rows: 300px 180px 230px;  
}
```



результат

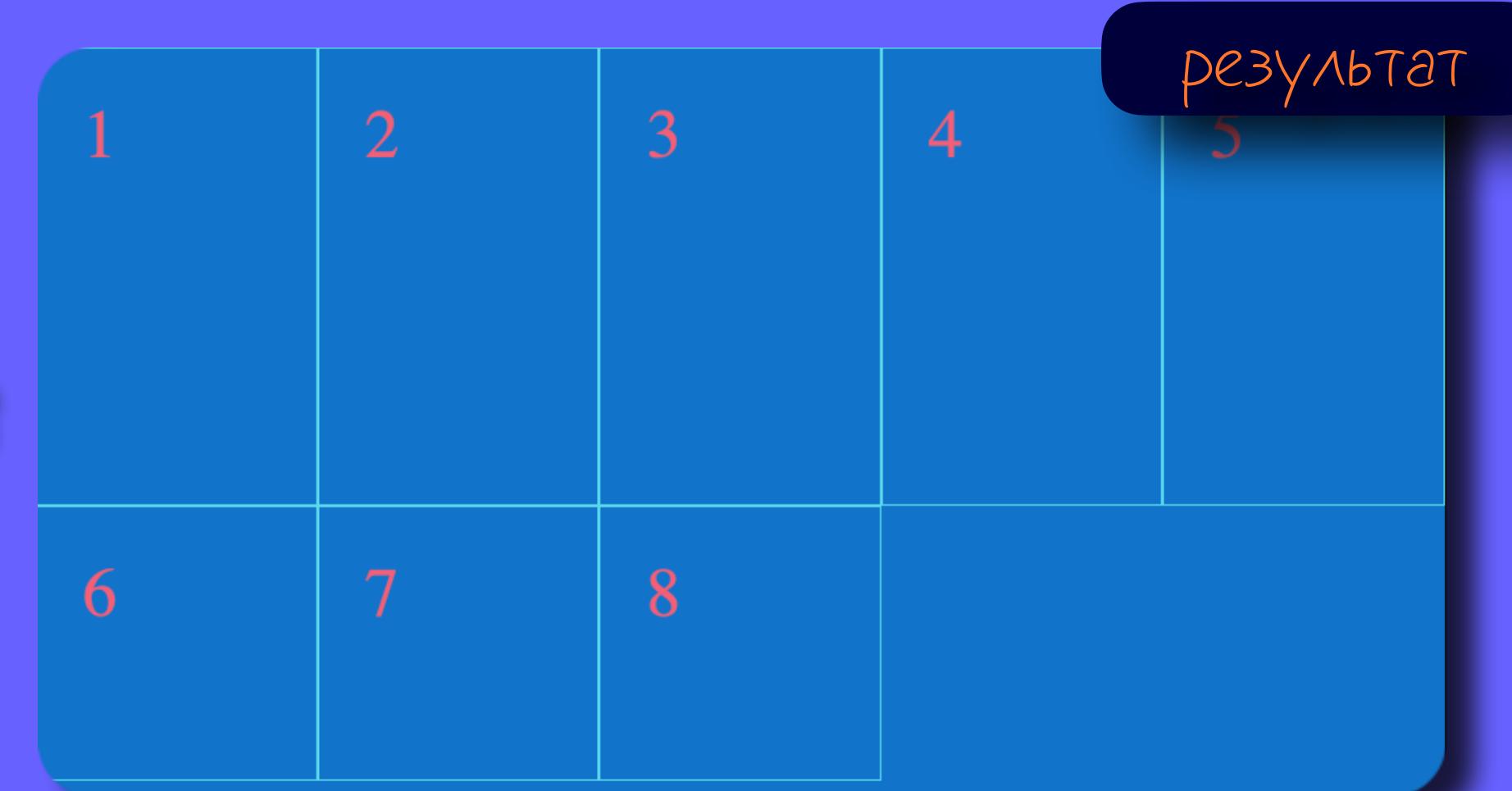
1	2	3	4
5	6	7	8

% VS fr

Вычислять процентное соотношение, для того чтобы вместить определенное количество колонок может быть накладным, поэтому на помощь здесь приходит *flexible track* - гибкие дорожки

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: repeat(5, 1fr);  
  grid-template-rows: 300px 180px 230px;  
}
```

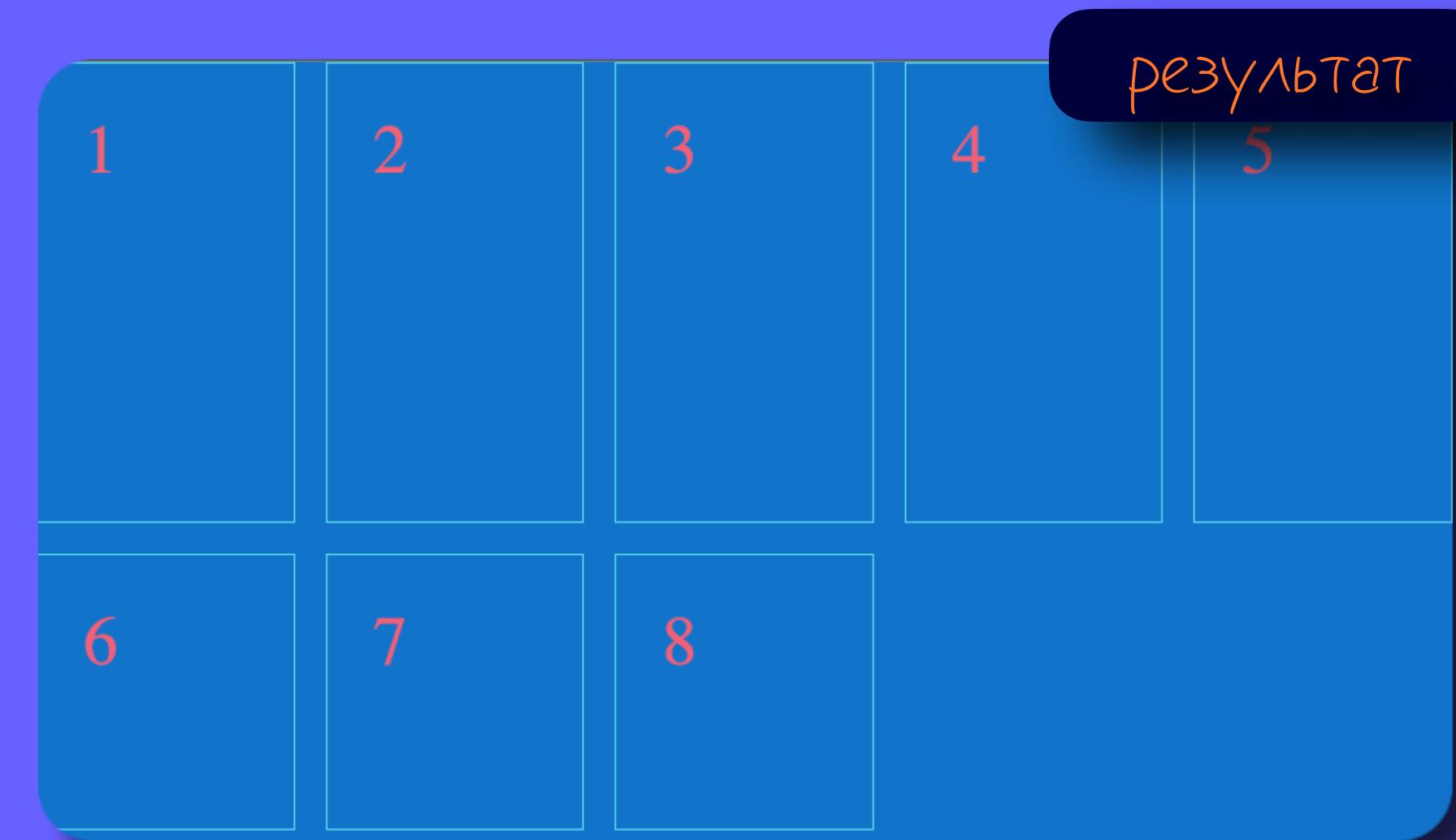


Gap - отступы

Также можно воспользоваться универсальным свойством `gap`, чтобы определить отступы между колонками и рядами.

`index.scss`

```
.grid-wrap {  
    display: grid;  
    grid-template-columns: repeat(5, 1fr);  
    grid-template-rows: 300px 180px 230px;  
    gap: 20px;  
}
```



Заметьте!!! Если бы здесь для `grid-template-columns` вы использовали бы процентное соотношение, то вам необходимо было бы и учитывать эти меж интервальные отступы. Иначе используя единицы измерения `fr` браузер гибко высчитывает все сам.

row-gap и column-gap

index.scss

```
.grid-wrap {
  display: grid;
  grid-template-columns: repeat(5, 1fr);
  grid-template-rows: 300px 180px 230px;
  row-gap: 20px;
  column-gap: 60px;
}
```

результат

1	2	3	4	5
6	7	8		

Оператор minmax()

index.scss

```
.grid-wrap {  
    display: grid;  
    grid-template-columns: 200px minmax(150px, 1fr) 200px;  
    grid-template-rows: 300px 180px 230px;  
    row-gap: 20px;  
    column-gap: 60px;  
}
```

Оператор `minmax(min, max)` – задаст минимальное значение для сжатия элемента и максимальное значение при увеличении элемента.

Заметьте!!! Что минимальное значение нельзя задавать в единицах измерения `fr` – это поломает поведения вашего `grid` контейнера. Единица `fr` здесь может выступать только в качестве максимального значения.

auto-fill

Вместо того, чтобы указывать количество колонок и сколько раз им повторяться, мы просто можем сказать браузеру, чтобы он уместил как можно больше колонок с учетом указанной длины.

auto-fill как бы говорит “я автоматически заполню строку таким количеством колонок, как это возможно с учетом заданной ширины”. auto-fill используется в связке с repeat() таким образом:

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: repeat(auto-fill, 400px);  
  grid-template-rows: 1fr;  
  justify-content: center;  
}
```



результат

1	2
3	4
5	6
7	8

auto-fit

Разницы auto-fit от auto-fill почти не ощущается, если не применить их в связке с оператором minmax:

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
  grid-template-rows: 1fr;  
}
```

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

результат

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));  
  grid-template-rows: 1fr;  
}
```

VS

Как видно auto-fit будет полностью
заполнять контейнер в отличии от auto-fill,
который оставляет пустое пространство.

результат

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

Auto

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: auto repeat(2, minmax(200px, 1fr));  
  grid-template-rows: 1fr;  
}
```

Лорем ипсум, dolor sit amet consectetur adipisicing elit. Nesciunt voluptates velit earum dicta, vero rerum assumenda voluptatem, nam explicabo saepe atque fugiat repudiandae! Non ipsam laborum similique est sapiente necessitatibus.

2

3

4

5

6

7

8

результат

Авто будет выставлять размер ячейки на максимальную ширину относительно своего наполнения (содержимого)

fit-content

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: fit-content(700px) repeat(2, minmax(200px, 1fr));  
  grid-template-rows: 1fr;  
}
```

Lorem ipsum, dolor sit amet
consectetur adipisicing elit.
Nesciunt voluptates velit earum
dicta, vero rerum assumenda
voluptatem, nam explicabo saepe
atque fugiat repudiandae! Non
ipsam laborum similique est
sapiente necessitatibus.

2

3

4

5

6

7

8

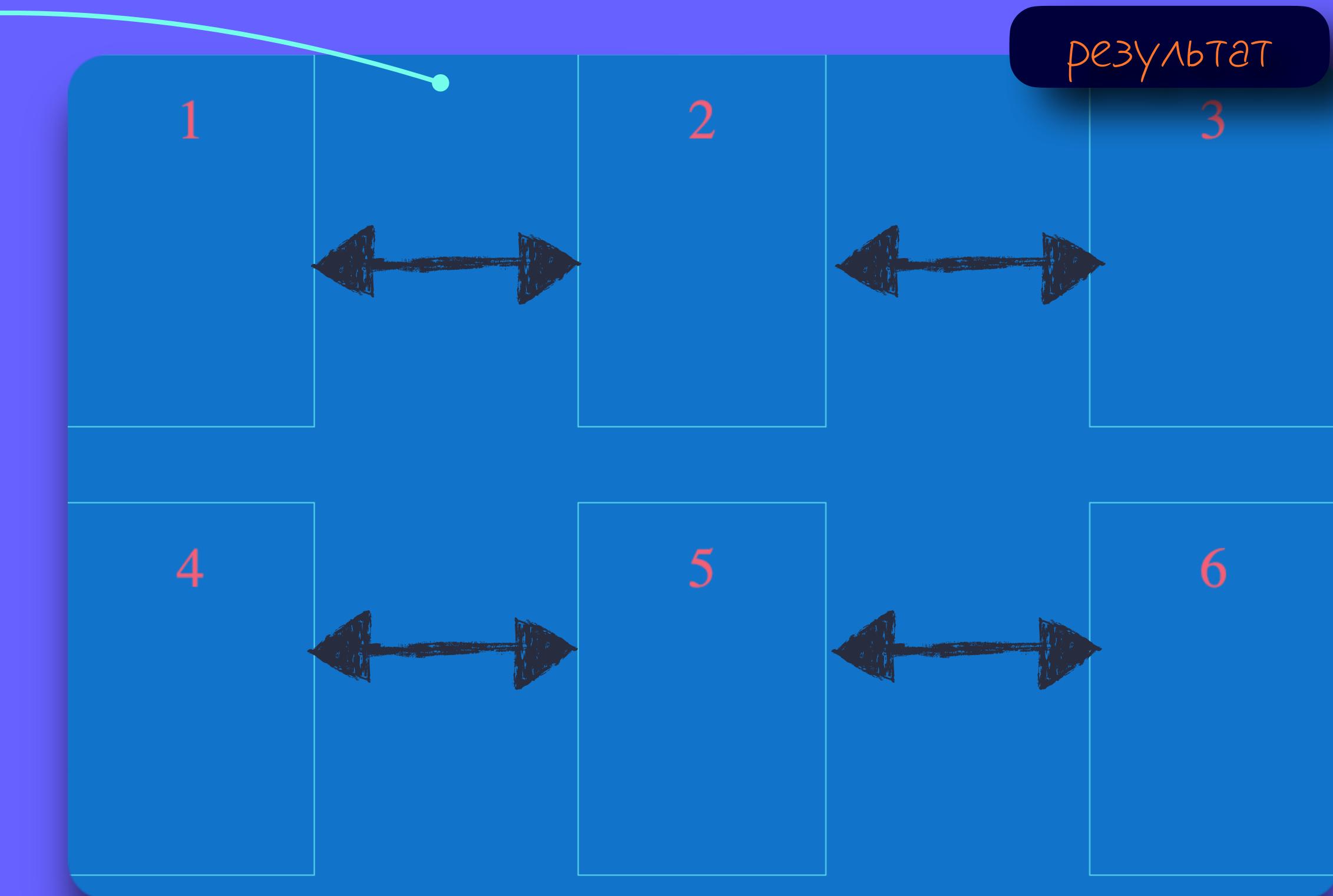
результат

fit-content(value) будет заполнять ячейку на заданное значение, при этом имея возможность сжиматься если не хватает ширины контейнера.

Justify-content

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: repeat(3, minmax(100px, 200px));  
  grid-template-rows: repeat(3, 300px);  
  justify-content: space-between;  
  gap: 20px;  
  column-gap: 40px;  
  row-gap: 60px;  
}
```



Justify-content – распределение элементов по горизонтали, работает таким же способом как и с
display: flex.

Align-items

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: repeat(3, minmax(100px, 200px));  
  grid-template-rows: repeat(3, 300px);  
  justify-content: space-between;  
  align-items: center;  
  gap: 20px;  
  column-gap: 40px;  
  row-gap: 60px;  
}
```



Align-items – распределение элементов по вертикали внутри ячейки.

Align-content

index.scss

```
.grid-wrap {  
  display: grid;  
  grid-template-columns: repeat(3, minmax(100px, 200px));  
  grid-template-rows: repeat(3, 100px);  
  justify-content: space-between;  
  align-items: center;  
  align-content: space-between;  
  gap: 20px;  
  column-gap: 40px;  
  row-gap: 60px;  
  height: 100vh;  
}
```

The diagram shows a 3x3 grid of boxes labeled 1 through 8. The columns are labeled 1, 2, and 3 from left to right. The rows are labeled 1, 2, and 3 from top to bottom. Each box contains a red number. Vertical double-headed arrows between the boxes indicate the distribution of vertical space within each column. A green dot marks the center of the grid. Arrows point upwards from box 1 to box 4, downwards from box 4 to box 7, upwards from box 7 to box 8, and downwards from box 8 back to box 1, forming a cycle. This visualizes how 'align-content: space-between;' distributes vertical space between the grid items.

Align-content – распределение элементов по вертикали внутри контейнера. Поэтому обрати внимание, что мы сначала сделали пространство для этого внутри контейнера в `height: 100vh`.

ПРАКТИКА

hlegal
LAW COMPANY

УК РУ EN • Kyiv, Mechnikova str, 14/1 • 0800 211 927 • 

[about us](#) [services](#) [team](#) [publications](#) [contact](#)

you legal solutions provider



Создание фона

index.scss

```
.section-gradient {
  background: linear-gradient(
    228.57deg,
    #323264 11.93%,
    #323264 57.17%,
    #643e72 87.46%
  );
  border-radius: 0px 0px 12px 12px;
  min-height: 100vh;
  margin-bottom: 20px;
  position: relative;
  &__shadow {
    position: absolute;
    height: 10px;
    border-radius: 0px 0px 12px 12px;
  }
  &__shadow--1 {
    bottom: -10px;
    width: calc(100% - 40px);
    left: 20px;
    background: #cacadb;
  }
  &__shadow--2 {
    bottom: -20px;
    width: calc(100% - 80px);
    left: 40px;
    background: #e7e7f2;
  }
}
```

index.html

```
<body>
  <div class="section-gradient">
    <div class="section-gradient__shadow section-gradient__shadow--1"></div>
    <div class="section-gradient__shadow section-gradient__shadow--2"></div>
  </div>
</body>
```

результат



Создание фона

index.scss

```
main-wrap {  
  min-height: 100vh;  
  position: relative;  
}  
  
main-wrap-bg {  
  min-height: 100%;  
  width: 100%;  
  position: absolute;  
  display: flex;  
  align-items: stretch;  
  justify-content: center;  
  overflow-x: hidden;  
  &__object {  
    position: absolute;  
    height: 100%;  
  }  
}
```

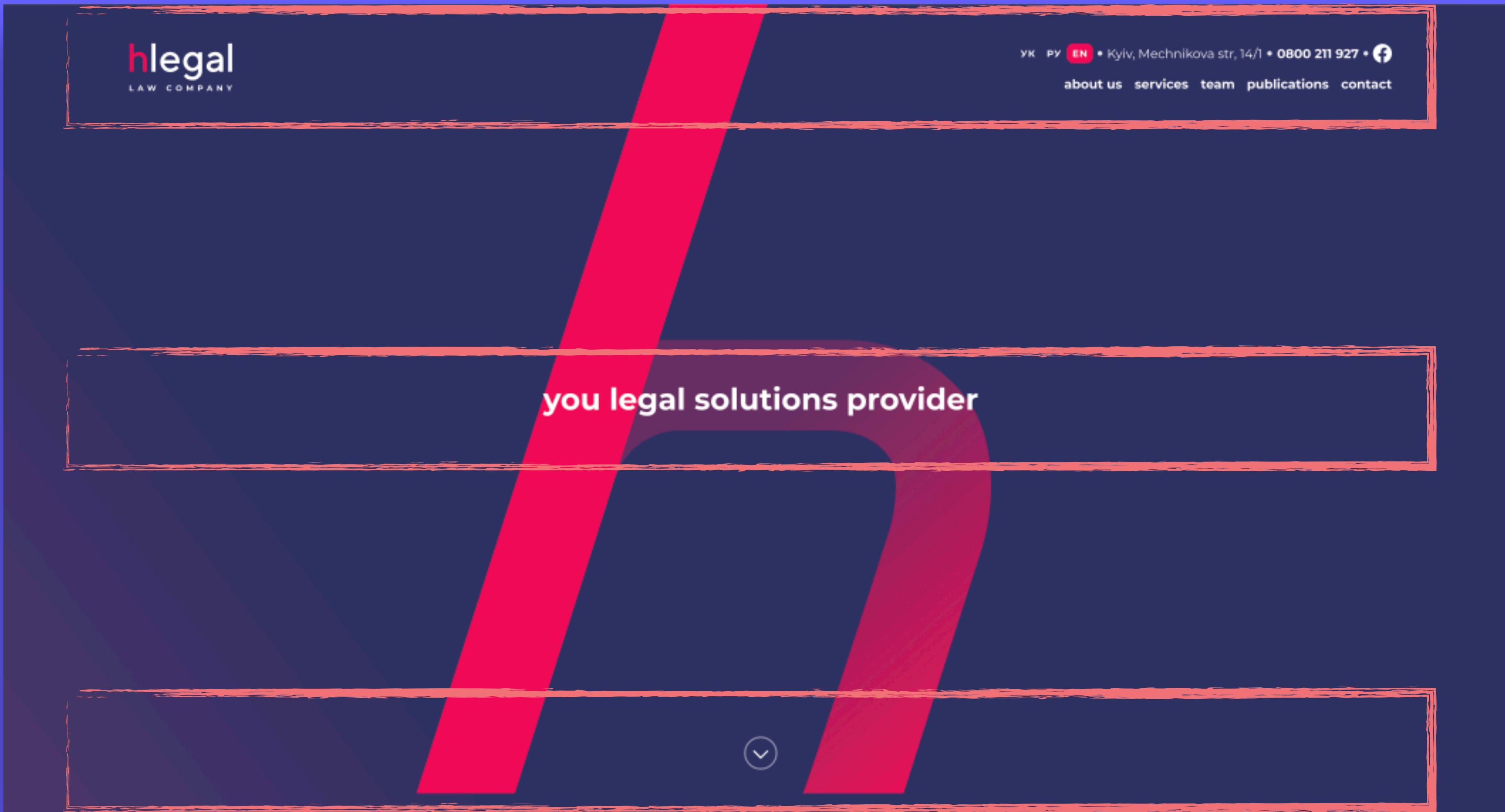
index.html

```
<div class="section-gradient">  
  <div class="section-gradient__shadow section-gradient__shadow--1"></div>  
  <div class="section-gradient__shadow section-gradient__shadow--2"></div>  
  <div class="main-wrap">  
    <div class="main-wrap-bg">  
      <object  
        class="main-wrap-bg__object"  
        data=".assets/bgi/h_logo.svg"  
        type=""></object>  
    </div>  
  </div>
```

результат



Распределение блоков



Распределение блоков

index.scss

```
.section-first {
  min-height: 100vh;
  position: relative;
  z-index: 100;
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  &_header,
  &_title,
  &_button {
    padding: 100px;
    text-align: center;
    font-size: 26px;
    color: white;
  }
  &_header {
    border: 3px solid white;
  }
  &_title {
    border: 3px solid red;
  }
  &_button {
    border: 3px solid black;
  }
}
```

index.html

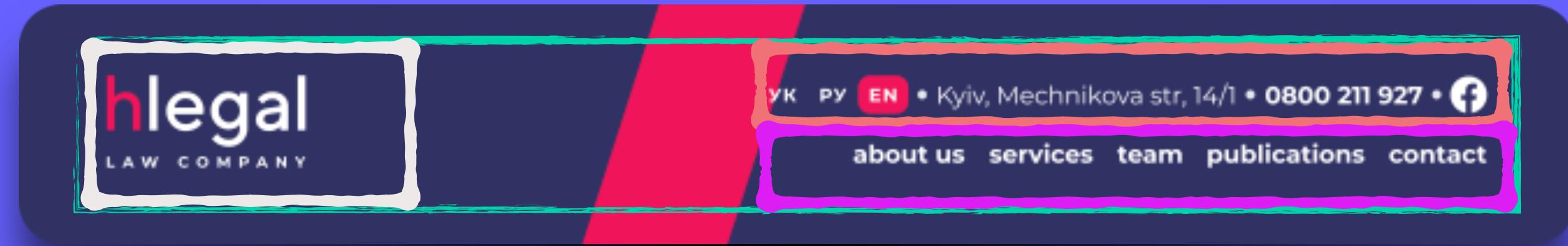
```
<div class="section-gradient">
  <div class="section-gradient__shadow section-gradient__shadow--1"></div>
  <div class="section-gradient__shadow section-gradient__shadow--2"></div>
  <div class="main-wrap">
    <div class="main-wrap-bg">
      <object
        class="main-wrap-bg__object"
        data="./assets/bgi/h_logo.svg"
        type=""
      ></object>
    </div>
    <div class="section-first">
      <header class="section-first__header">Header</header>
      <main class="section-first__title">Title</main>
      <div class="section-first__button">Button</div>
    </div>
  </div>
</div>
```

результат



Создание адаптивного хедера

Grid



Как видно из дизайна у нас содержимое хедера располагается в два ряда – где с лева это логотип а с права у нас контактная информация и навигация.



Но затем на планшете контактная информация становится сверху, логотип по середине, а навигация снизу.

Создание адаптивного хедера

index.scss

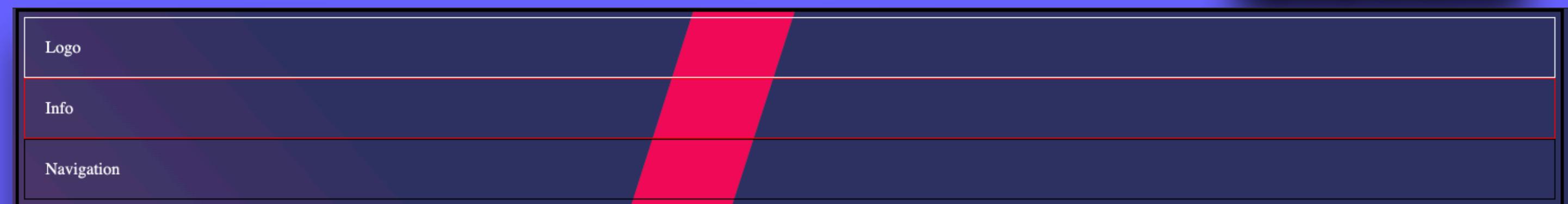
```
.header-grid {  
    border: 3px solid black;  
    padding: 5px;  
    color: white;  
    &__logo,  
    &__info,  
    &__nav {  
        padding: 20px;  
    }  
    &__logo {  
        border: 1px solid white;  
    }  
    &__info {  
        border: 1px solid red;  
    }  
    &__nav {  
        border: 1px solid black;  
    }  
}
```

index.html

```
<div class="section-first">  
    <header class="section-first__header">  
        <div class="container">  
            <div class="header-grid">  
                <div class="header-grid__logo">Logo</div>  
                <div class="header-grid__info">Info</div>  
                <nav class="header-grid__nav">Navigation</nav>  
            </div>  
        </div>  
    </header>  
    <main class="section-first__title"></main>  
    <div class="section-first__button"></div>  
</div>
```

результат

Создадим приблизительный вид и структуры header, а
далее попробуем применить `display: grid` для
достижения необходимого результата.



Grid-template-areas и grid-area

index.scss

```
.header-grid {  
    border: 3px solid black;  
    padding: 5px;  
    color: white;  
    display: grid;  
    grid-template-areas:  
        "logo info"  
        "logo nav";  
    justify-content: space-between;  
  
    @media screen and (max-width: 768px) {  
        grid-template-columns: 1fr;  
        grid-template-areas:  
            "info"  
            "logo"  
            "nav";  
    }  
}  
&_logo,  
&_info,  
&_nav {  
    padding: 20px;  
}  
&_logo {  
    border: 1px solid white;  
    grid-area: logo;  
}  
&_info {  
    border: 1px solid red;  
    grid-area: info;  
}  
&_nav {  
    border: 1px solid black;  
    grid-area: nav;  
}
```

результат



Grid-template-areas позволяет нам указать как будет располагаться в сетке (grid) каждая ячейка по имени.



Grid-area укажет имя ячейки для данного селектора.