

## Практическая работа №8 «CSS Grid Layout»

### 17 Grid Layout

**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 (строками) и вертикалью (колонками). **Flexbox** это одномерные макеты, а **Grid** это двумерные.

В концепциях CSS Grid на первом месте стоит сам макет, а во Flexbox акцент делается на содержимом.

Flexbox-макет отлично подходит для компонентов приложения и маломасштабных макетов. Grid-макет подходит для более масштабных макетов с нелинейным дизайном.

Принцип работы гридов чем-то похож на таблицы. Вместо работы только с рядами или только с колонками с помощью гридов можно работать с так называемыми грид-ячейками, позиционируя элементы по вертикали и горизонтали одновременно.

Ниже будут описаны основные используемые свойства гридов; если потребуется информация, которой не будет представлено, вы можете обратиться к спецификации:

Спецификация CSS Grid Layout – URL: <https://www.w3.org/TR/css-grid-1/>.

#### 17.1 Основные понятия Grid

1. **Grid-контейнер** - родительский элемент, к которому применяется свойство `display: grid`.
2. **Grid-элемент** - дочерний элемент, прямой потомок grid-контейнера. Подчиняется правилам раскладки гридов.
3. **Grid-линия** - разделительная линия, формирующая структуру грида (рисунок 17.1). Может быть как вертикальной (grid-линия колонки), так и горизонтальной (grid-линия ряда). Располагается по обе стороны от колонки или ряда. Используется для привязки grid-элементов.

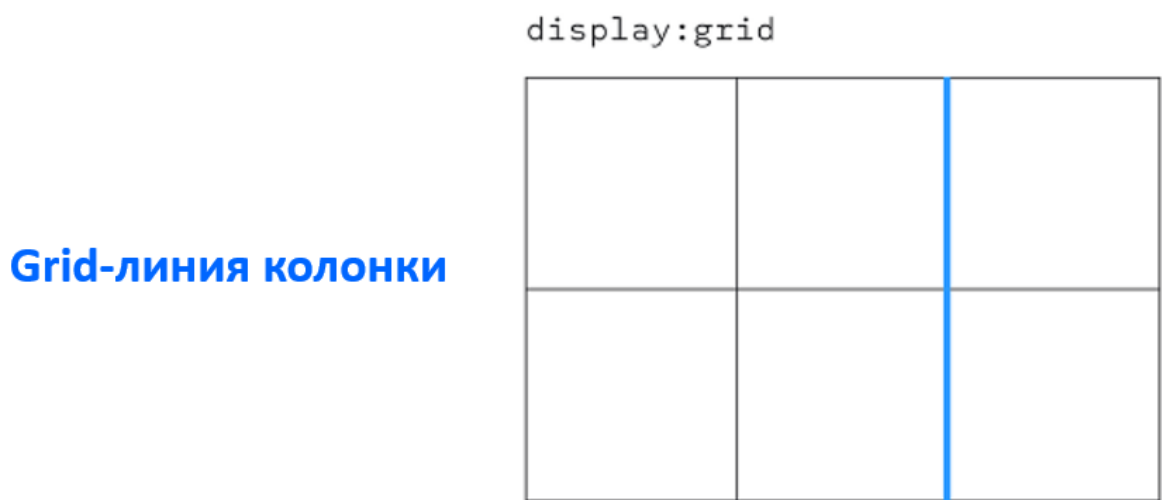


Рисунок 17.1 - Grid-линия колонки

4. **Grid-ячейка** - пространство между соседними grid-линиями (рисунок 17.2).  
Единица grid-сетки.

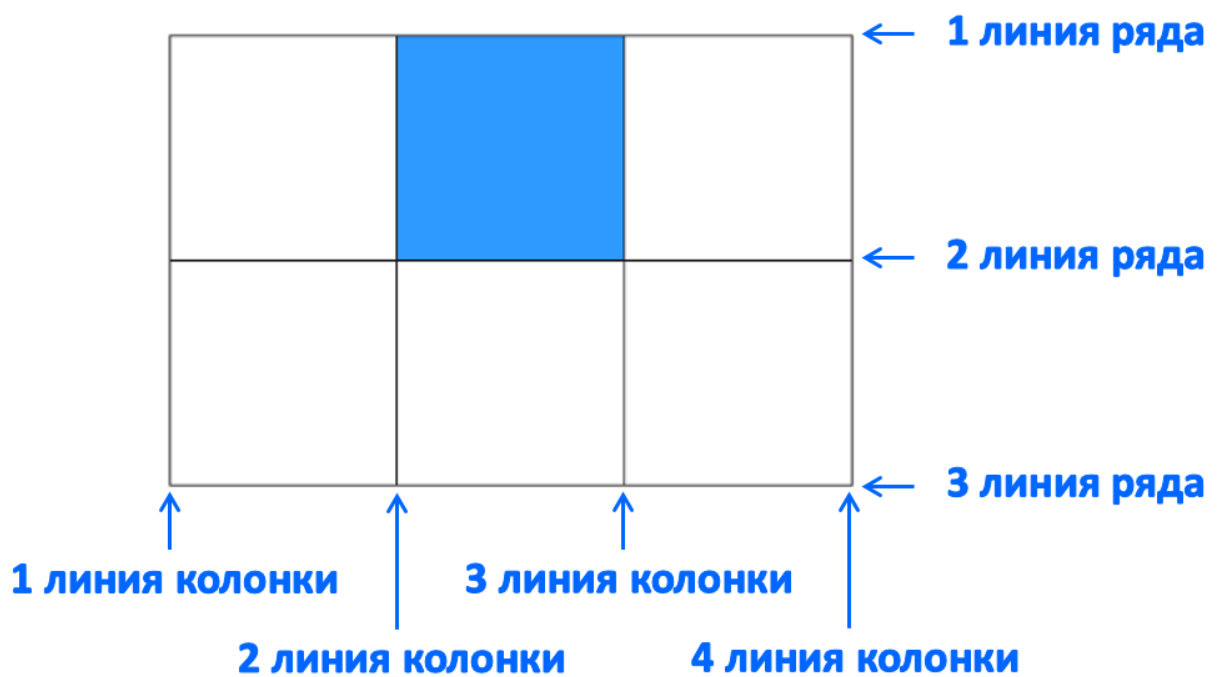


Рисунок 17.2 - Grid-ячейка между первой и второй grid-линиями ряда и второй и третьей grid-линиями колонки

5. **Grid-полоса** - пространство между двумя соседними grid-линиями (рисунок 17.3).

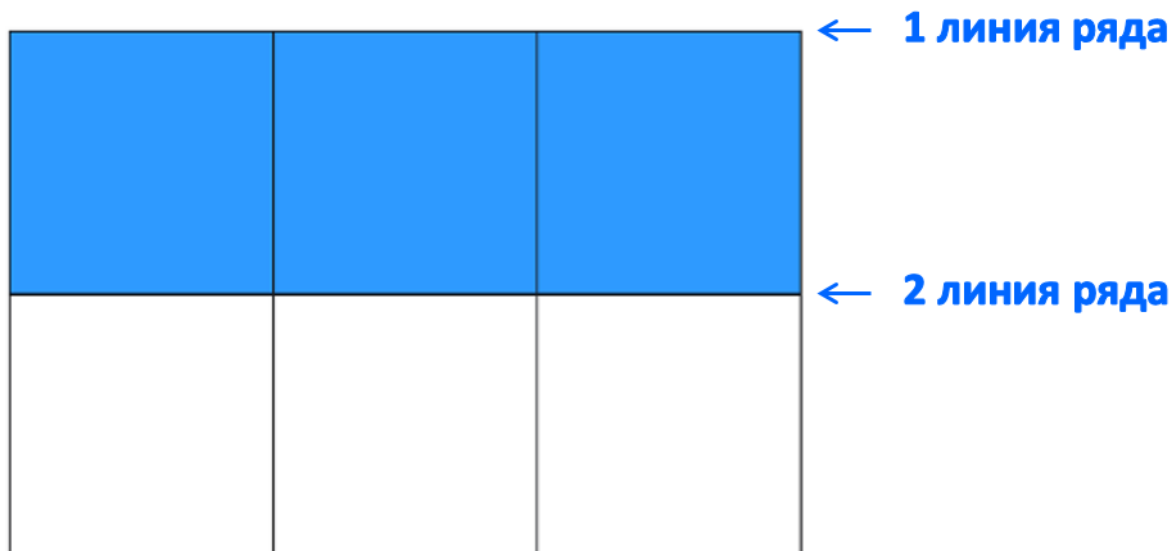


Рисунок 17.3 - Горизонтальная grid-полоса между первой и второй grid-линиями ряда

- 6. Grid-область** - область, ограниченная четырьмя grid-линиями (рисунок 17.4).  
 Может состоять из любого количества ячеек как по вертикали, так и по горизонтали.

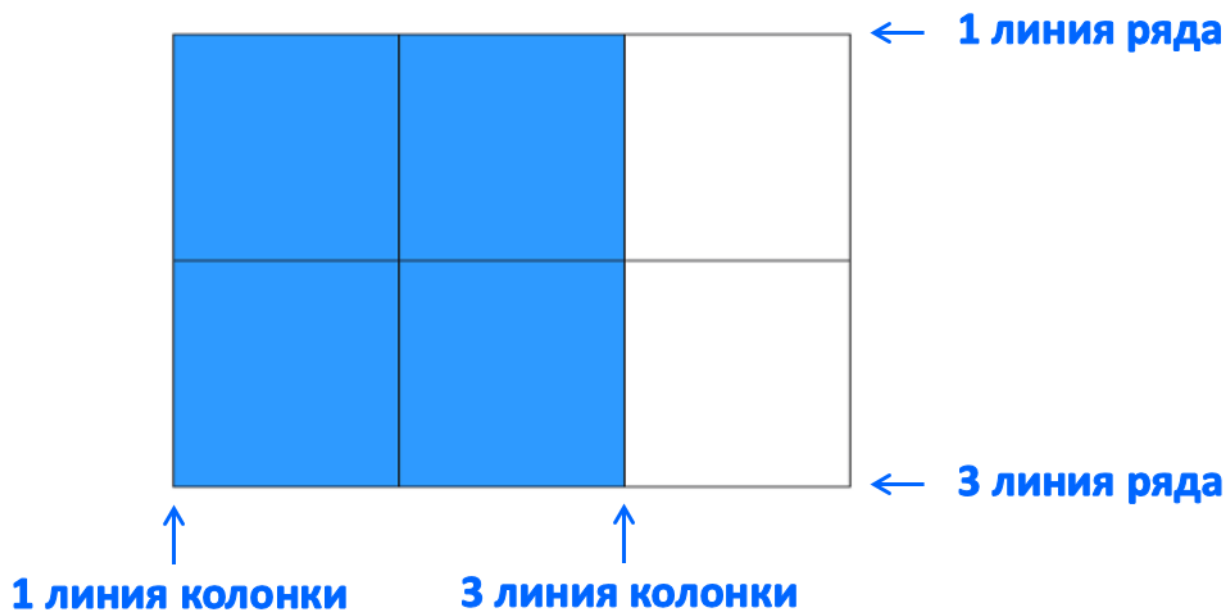


Рисунок 17.4 - Grid-область между первой и третьей grid-линиями ряда и первой и третьей grid-линиями колонки

## 17.2 Свойства grid-контейнера

### 17.2.1 Свойство display

Если элементу задано свойство `display` со значением `grid`, то такой элемент становится грид-контейнером (листинг 17.2.1). Дочерние элементы этого контейнера начинают подчиняться правилам грид-раскладки. Снаружи грид-контейнер ведёт себя как блок.

Листинг 17.2.1 – Значение `grid`

```
.container {  
  display: grid;  
}
```

Значение `inline-grid` практически аналогично предыдущему - за тем исключением, что в этом случае грид-контейнер снаружи будет вести себя как строчный элемент (листинг 17.2.2).

Листинг 17.2.2 - Значение `inline-grid`

```
.container {  
  display: inline-grid;  
}
```

### 17.2.2 Свойства `grid-template-columns` и `grid-template-rows`

Свойства, задающие размеры и количество колонок или рядов грид-раскладки (листинг 17.2.3 и рисунок 17.2.1).

Листинг 17.2.3 - Свойства `grid-template-columns` и `grid-template-rows`

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

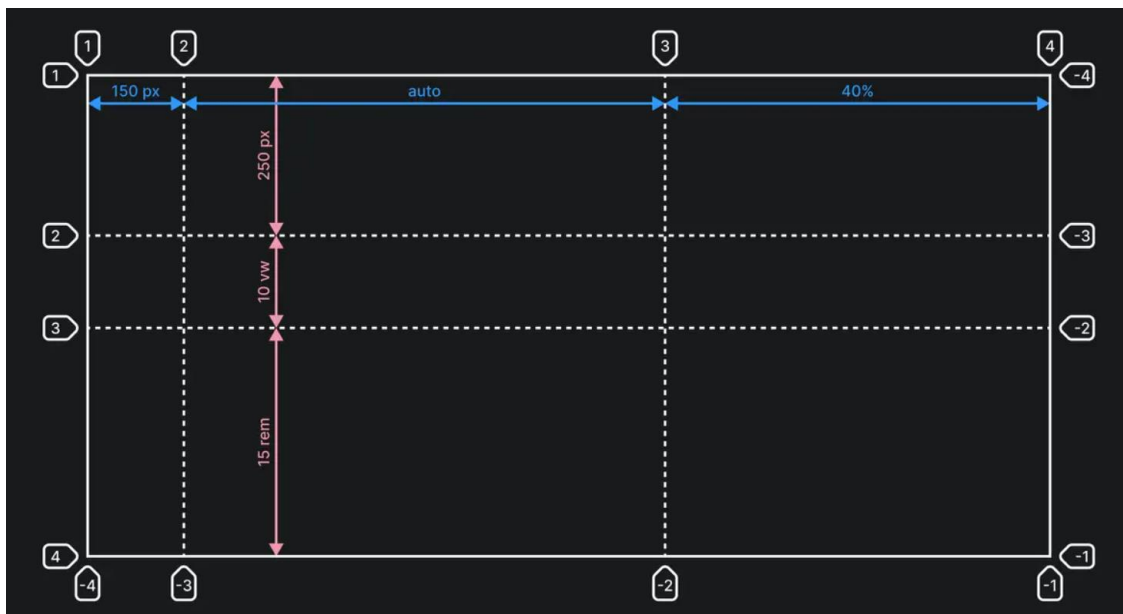


Рисунок 17.2.1 - Свойства grid-template-columns и grid-template-rows

Вы можете также явно именовать грид-линии, используя для этого квадратные скобки (листинг 17.2.4, рисунок 17.2.2). Каждая линия также может иметь больше одного имени – они пишутся через пробел.

Листинг 17.2.4 – Именованное grid-линий

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

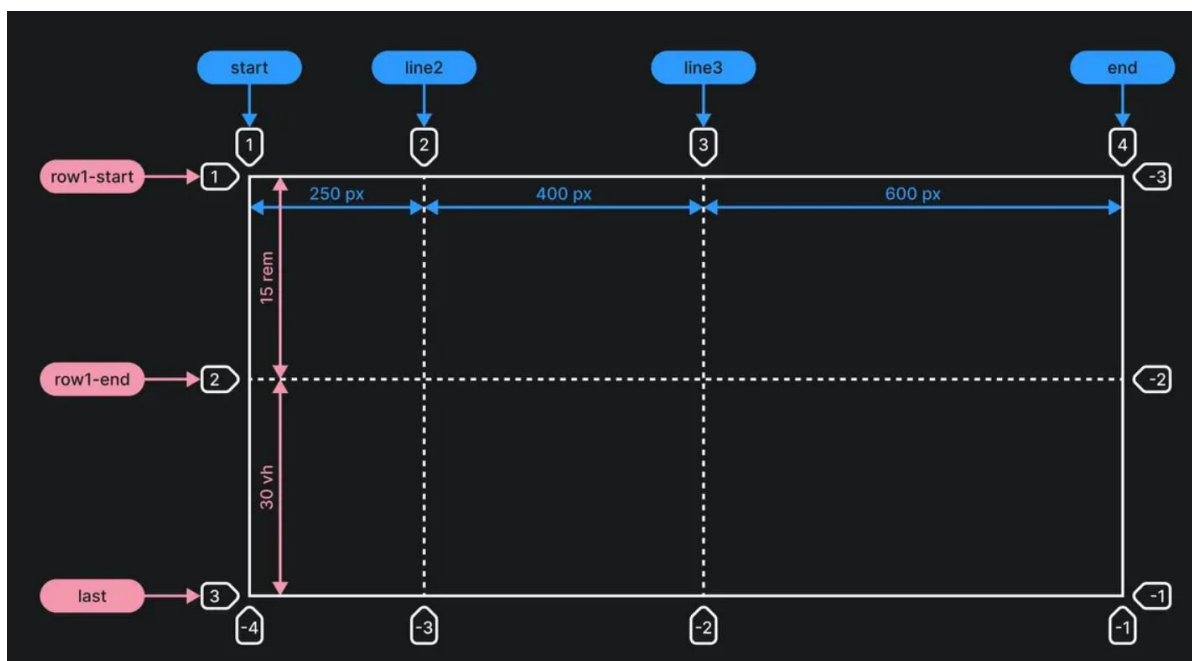


Рисунок 17.2.2 – Именованное grid-линий

Если нужны одинаковые колонки или ряды, то можно воспользоваться функцией `repeat()`. Пример ниже создает 3 колонки по 250 пикселей (листинг 17.2.5):

Листинг 17.2.5 – Именованное grid-линий

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

С появлением гридов появилась новая единица измерения: **fr**.

**fr** (от *fraction* — доля, часть) отвечает за свободное пространство внутри грид-контейнера. Например, код ниже создаст три колонки, каждая из которых будет занимать 1/3 ширины родителя (листинг 17.2.6):

Листинг 17.2.6 – Использование долей (fr) – сокращенное написание

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

Верхняя запись кода аналогична нижепредставленной (листинг 17.2.7):

Листинг 17.2.7 – Использование долей (fr) – полное написание

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

Свободное пространство рассчитывается после того, как место отдано всем фиксированным размерам. К примеру, в коде ниже сначала будет создана колонка шириной 250 пикселей, а затем свободное пространство — ширина родителя минус 250 пикселей — будет поделено между остальными колонками. Каждая будет занимать ширину  $(100\% - 250\text{px}) / 2$  (листинг 17.2.8):

Листинг 17.2.8 – Распределение свободного пространства

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 250px 1fr;  
}
```

### 17.2.3 Свойства `grid-auto-columns` и `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 (листинг 17.2.9 и рисунок 17.2.3).

Листинг 17.2.9 – Свойство `grid-auto-rows`

```
.container {  
  display: grid;  
  grid-template-rows: 50px 140px;  
  grid-auto-rows: 40px;  
  gap: 20px;  
}
```



Рисунок 17.2.3 – Свойство `grid-auto-rows`

В этом примере создаются два явных ряда размером 50 и 140 пикселей, соответственно. Элементы, начиная с третьего, в эти два ряда не помещаются, и для них создаются автоматические ряды. При помощи свойства `grid-auto-rows` мы указываем, что автоматические ряды должны иметь размер 40 пикселей.

Можно задавать больше одного значения для автоматических колонок или рядов. Тогда паттерн размера будет повторяться до тех пор, пока не кончатся грид-элементы.

### 17.2.4 Свойство `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 пустые ячейки в разметке, если размеры элементов позволяют. Можно сочетать с остальными значениями.

На примере, приведенном ниже, есть большой блок, который не помещается в одну грид-ячейку (рисунок 17.2.4 и листинг 17.2.10).

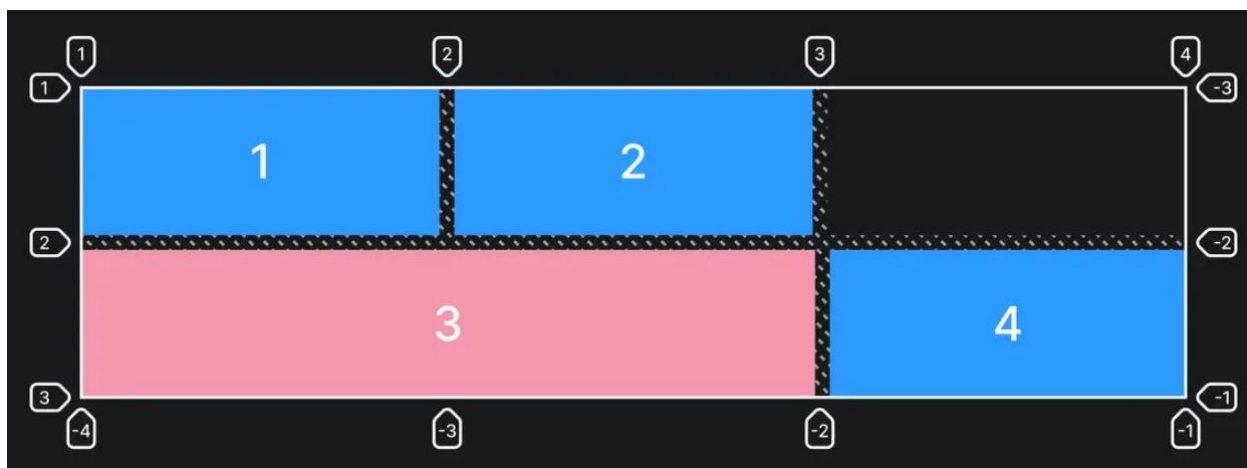


Рисунок 17.2.4 – Свойство `grid-auto-flow`

Как видно на рисунке 17.2.4, 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.



Листинг 17.2.10 – Свойство grid-auto-flow

```
.container {  
  display: grid;  
  
  /* Три колонки */  
  grid-template-columns: auto auto auto;  
  
  /* Два ряда */  
  grid-template-rows: auto auto;  
  
  /* Автоматическое размещение в ряд */  
  grid-auto-flow: row;  
  
  /* Отступы между ячейками */  
  gap: 20px;  
}  
  
.item3 {  
  /* Занимает один ряд и  
  растягивается на две колонки */  
  grid-column: span 2;  
}
```

Исправить данную ситуацию можно с помощью добавления значения `dense` к свойству `grid-auto-flow` (листинг 17.2.11 и рисунок 17.2.5):

Листинг 17.2.11 – Свойство grid-auto-flow со значением dense

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

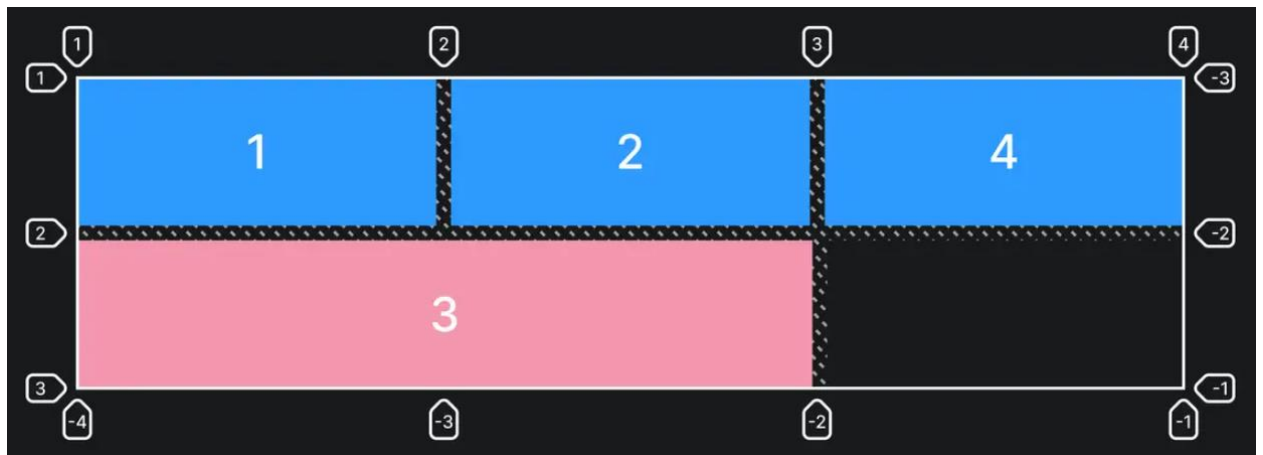


Рисунок 17.2.5 – Свойство grid-auto-flow со значением dense

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

С колонками (`grid-auto-flow: column;`) значение `dense` работает аналогично.

### 17.2.5 Свойство `grid-template-areas`

Свойство `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. Удобнее всего будет подписывать области в виде некой таблицы (листинг 17.2.12 и рисунок 17.2.6).

**Имена областей должны разделяться пробелами.** Это важно, особенно в том случае, если вы хотите расположить две пустых ячейки рядом. Разделите точки пробелами, иначе браузер подумает, что это одна пустая ячейка.

### Листинг 17.2.12 – Свойство `grid-template-areas`

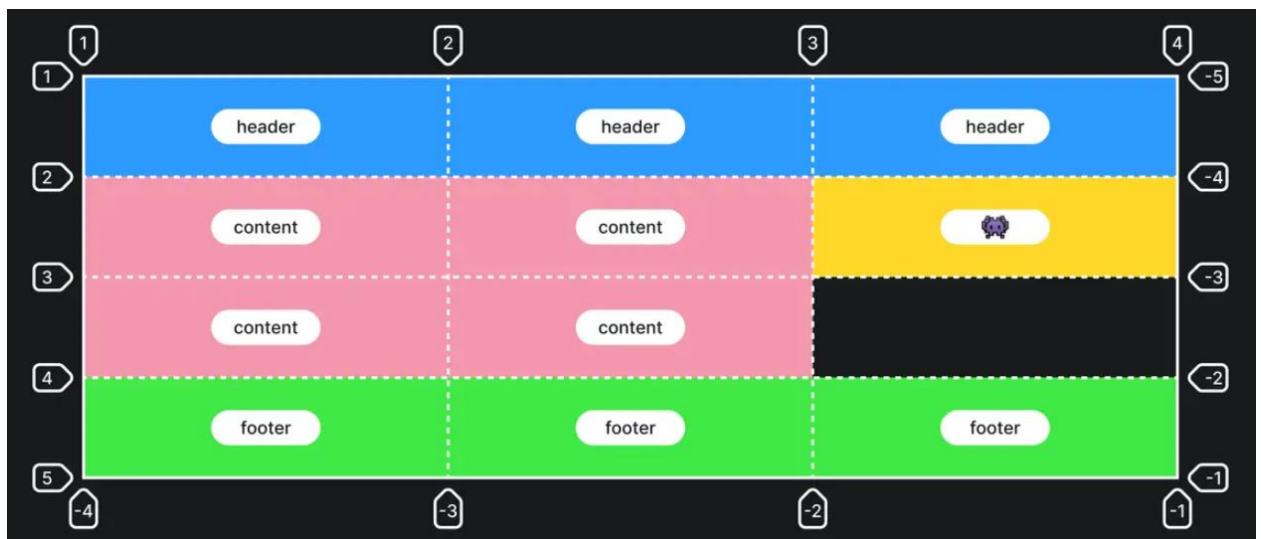


Рисунок 17.2.6 - Свойство `grid-template-areas`

Собирающее свойство (свойство-шорткат) для свойств `grid-template-rows`, `grid-template-columns`. Позволяет записать все значения в одну строку.

### Листинг 17.2.13 – Свойство grid-template

```
.container {  
  display: grid;  
  grid-template: repeat(4, 1fr) / repeat(3, 500px);  
}
```

### 17.2.7 Свойства column-gap и row-gap

Свойства `column-gap` и `row-gap` задают отступы между колонками или рядами (листинг 17.2.14 и рисунок 17.2.7).

### Листинг 17.2.14 – Свойства column-gap и row-gap

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

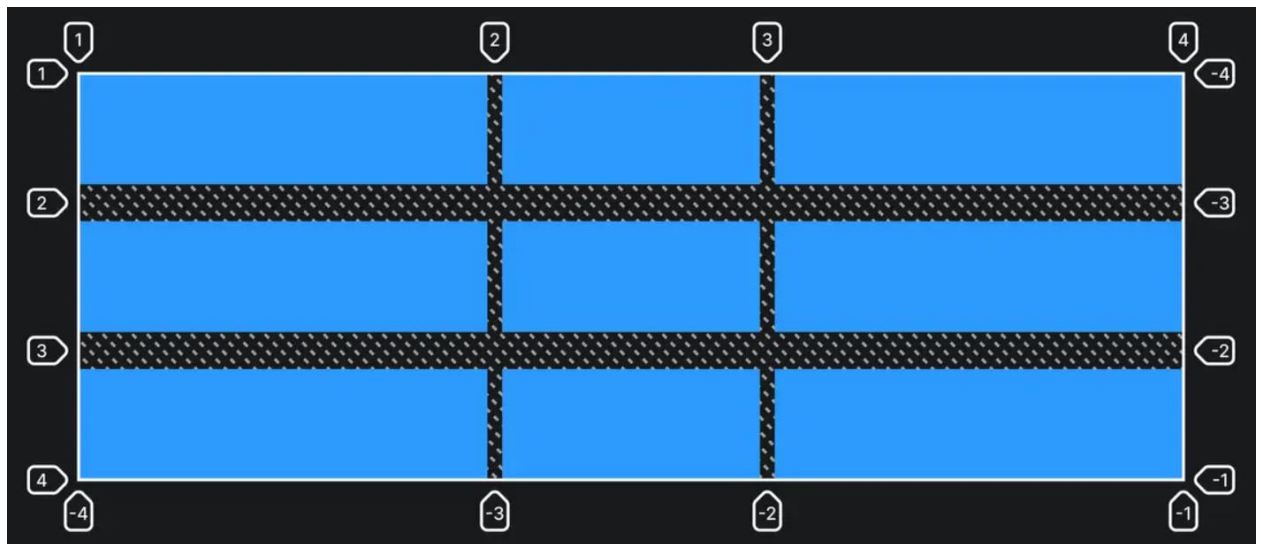


Рисунок 17.2.7 - Свойства column-gap и row-gap

### 17.2.8 Свойство gap

Свойство-шорткат для записи значений свойств `row-gap` и `column-gap`. Значения разделяются пробелом (листинг 17.2.15):

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

### 17.2.9 Свойство justify-content

Свойство `justify-content` позволяет 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.

Возможные значения:

- `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.

### 17.2.10 Свойство justify-items

Свойство `justify-items` задает выравнивание грид-элементов по горизонтальной оси. Применяется ко всем элементам внутри грид-родителя.

Возможные значения:

- `start` - выравнивает элемент по начальной (левой) линии (листинг 17.2.16 и рисунок 17.2.8).

Листинг 17.2.16 – Свойство justify-items со значением start

```
.item {
  min-width: 150px;
}

.container {
  display: grid;
  grid-template-columns: 1fr 200px 1fr;
  grid-template-rows: repeat(3, 150px);
  gap: 20px;

  justify-items: start;
}
```

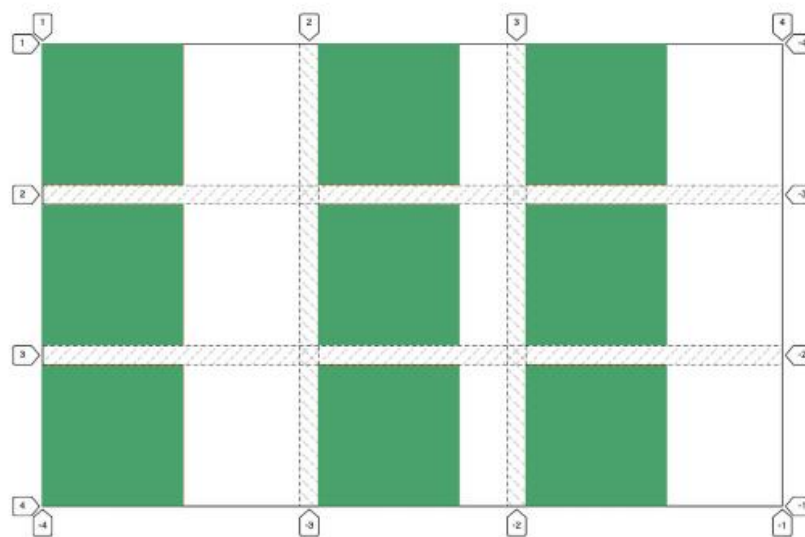


Рисунок 17.2.8 - Свойство justify-items со значением start

- **end** - выравнивает элемент по конечной (правой) линии (листинг 17.2.17 и рисунок 17.2.9).

Листинг 17.2.17 – Свойство justify-items со значением end

```
.container {
  display: grid;
  grid-template-columns: 1fr 200px 1fr;
  grid-template-rows: repeat(3, 150px);
  gap: 20px;

  justify-items: end;
}
```

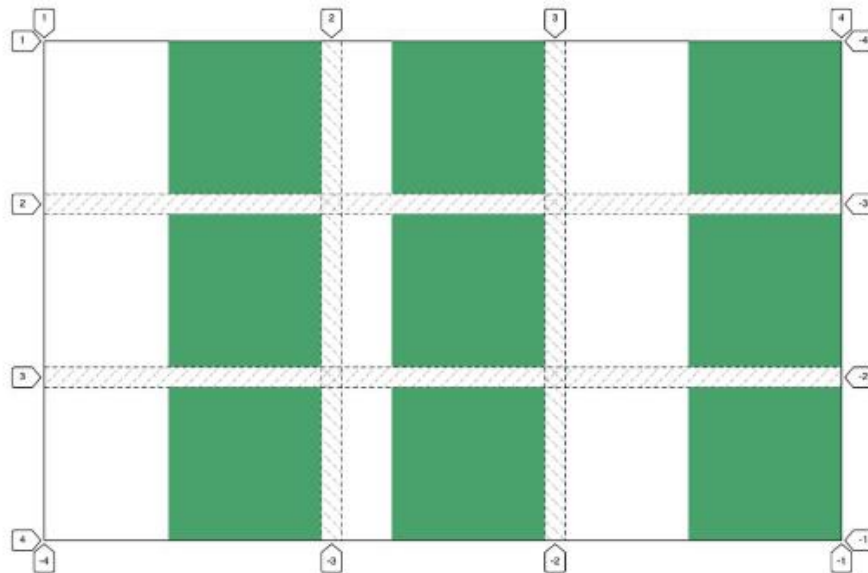


Рисунок 17.2.9 - Свойство justify-items со значением end

- **center** - выравнивает элемент по центру грид-ячейки (листинг 17.2.18 и рисунок 17.2.10).

Листинг 17.2.18 – Свойство justify-items со значением center

```
.container {
  display: grid;
  grid-template-columns: 1fr 200px 1fr;
  grid-template-rows: repeat(3, 150px);
  gap: 20px;

  justify-items: center;
}
```

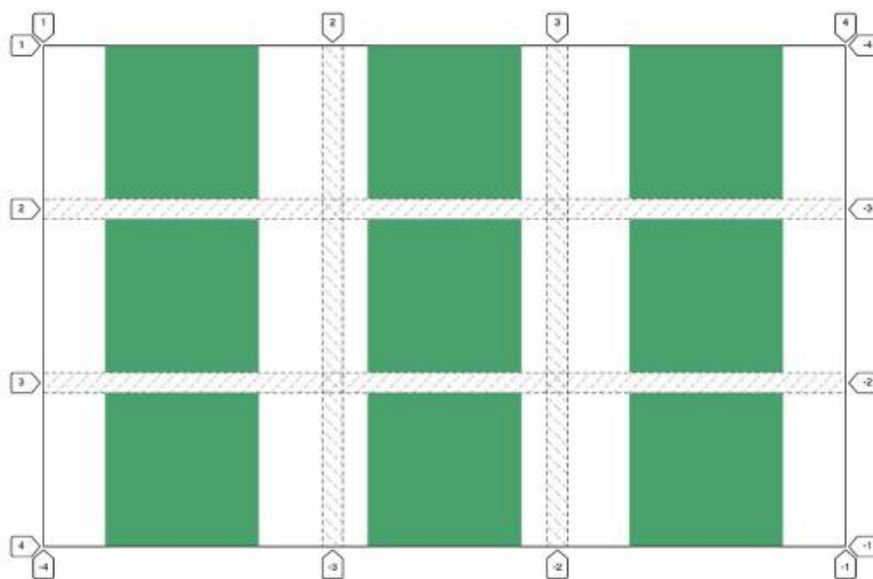


Рисунок 17.2.10 - Свойство justify-items со значением center

- **stretch** - растягивает элемент на всю ширину грид-ячейки (листинг 17.2.19 и рисунок 17.2.11).

Листинг 17.2.19 – Свойство `justify-items` со значением `stretch`

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 200px 1fr;  
  grid-template-rows: repeat(3, 150px);  
  gap: 20px;  
  
  justify-items: stretch;  
}
```

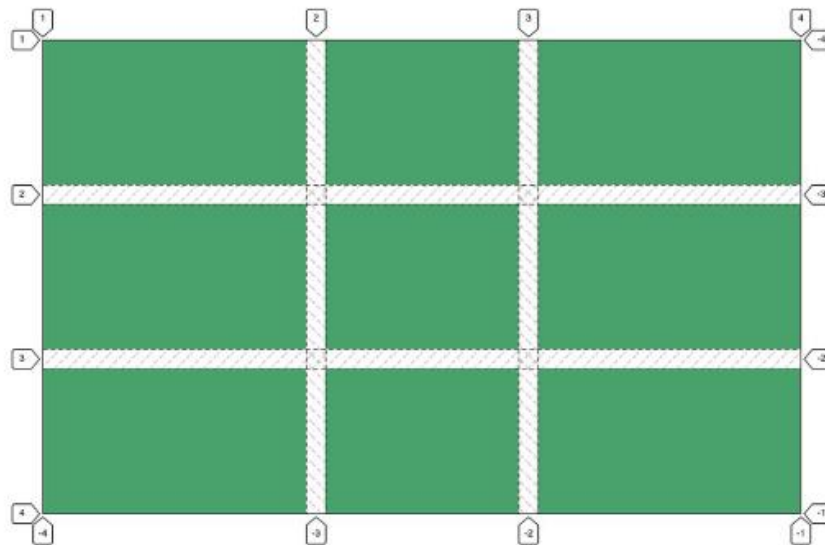


Рисунок 17.2.11 - Свойство `justify-items` со значением `stretch`

Можно управлять выравниванием отдельных грид-элементов при помощи свойства `justify-self`.

### 17.2.10 Свойство `align-items`

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

Возможные значения:

- **start** - выравнивает элемент по начальной (верхней) линии (листинг 17.2.20 и рисунок 17.2.12).



Листинг 17.2.20 – Свойство align-items со значением start

```
.item {
  min-height: 100px;
}

.container {
  display: grid;
  grid-template-columns: 1fr 200px 1fr;
  grid-template-rows: repeat(3, 150px);
  gap: 20px;

  align-items: start;
}
```

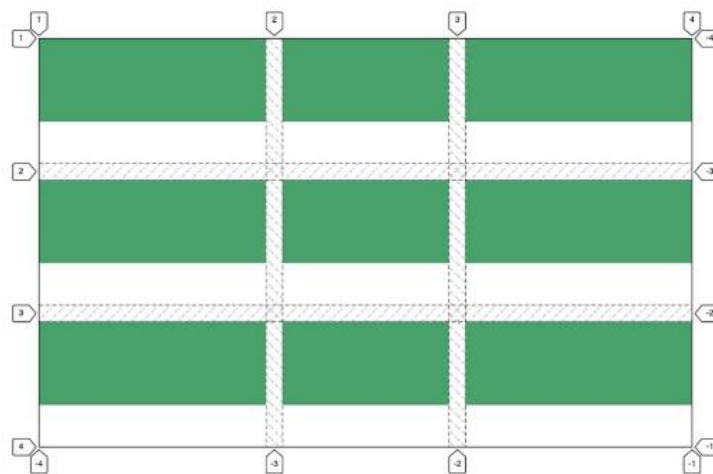


Рисунок 17.2.12 - Свойство align-items со значением start

- **end** - выравнивает элемент по конечной (нижней) линии.
- **center** - выравнивает элемент по центру грид-ячейки.
- **stretch** - растягивает элемент на всю высоту грид-ячейки.

Можно управлять выравниванием отдельных грид-элементов при помощи свойства `align-self`.

### 17.2.11 Свойство place-items

Свойство-шорткат для указания значений сразу и для `align-items` и для `justify-items` (листинг 17.2.21). Указывать нужно именно в таком порядке.

## Листинг 17.2.21 – Свойство place-items

```
.container {  
  display: grid;  
  place-items: stretch end;  
}
```

### 17.2.12 Свойство grid

Свойство, позволяющее задать значения нескольким свойствам сразу. А именно:

- `grid-template-rows`
- `grid-template-columns`
- `grid-template-areas`
- `grid-auto-rows`
- `grid-auto-columns`
- `grid-auto-flow`

**Прежде, чем использовать это объединяющее свойство, стоит несколько раз подумать о последующей читабельности кода.**

Возможные значения:

- `none` - значение по умолчанию. Это ключевое слово сбрасывает значения для всех свойств, входящих в это свойство.
- Можно указать допустимые значения для шортката `grid-template` (листинг 17.2.22):

## Листинг 17.2.22 - Допустимые значения для шортката grid-template

```
.container {  
  display: grid;  
  grid: repeat(4, 150px) / 1fr 200px 1fr;  
}
```

В том числе можно указать имена линий (листинг 17.2.23):

## Листинг 17.2.23 - Указание имен линий

```
.container {  
  display: grid;  
  grid:  
    [row1-start] 25px [row1-end row2-start] 25px [row2-end]  
    / auto 50px auto;  
}
```

Можно задать размеры колонок и рядов. Создадим два ряда и две колонки (листинг 17.2.24):

#### Листинг 17.2.24 – Создание двух рядов и двух колонок

```
.container {  
  display: grid;  
  grid: 200px 100px / 30% 30%;  
}
```

- `auto-flow` - это ключевое слово даёт браузеру понять, что создавать при необходимости: колонки или ряды. Если `auto-flow` стоит справа от слэша, то будут создаваться автоматические колонки (листинг 17.2.25), если слева, то – ряды (листинг 17.2.26).

#### Листинг 17.2.25 - Создание автоматических колонок

```
.container {  
  display: grid;  
  grid: 200px 100px / auto-flow 30%;  
}
```

#### Листинг 17.2.26 - Создание автоматических рядов

```
.container {  
  display: grid;  
  grid: auto-flow 30% / 200px 100px;  
}
```

- `dense` - к ключевому слову `auto-flow` можно добавить `dense`. Это укажет браузеру, что элементы должны стараться заполнить свободные ячейки. Важно использовать `dense` сразу после `auto-flow` (листинг 17.2.27).

#### Листинг 17.2.27 – Ключевое слово dense

```
.container {  
  display: grid;  
  grid: auto-flow dense 30% / 200px 100px;  
}
```

### 17.3 Свойства grid-элементов

**Примечание:** свойства `float`, `display: inline-block`, `display: table-cell`, `vertical-align` и `column-*` не дают никакого эффекта, когда применяются к грид-элементам.

### 17.3.1 Свойства `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 (листинг 17.3.1 и рисунок 17.3.1).

Возможные значения:

- **название** или **номер линии** - может быть порядковым номером или названием конкретной линии.
- **span число** - элемент растянется 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.

Листинг 17.3.1 - Свойства `grid-column-start`, `grid-column-end`, `grid-row-start`, `grid-row-end`

```
.container {  
  display: grid;  
  grid-template-columns: [one] 1fr [two] 1fr [three] 1fr [four] 1fr [five] 1fr [six];  
  grid-template-rows: [row1-start] 1fr [row1-end] 1fr 1fr;  
}  
  
.item1 {  
  grid-column-start: 2;  
  grid-column-end: five;  
  grid-row-start: row1-start;  
  grid-row-end: 3;  
}
```

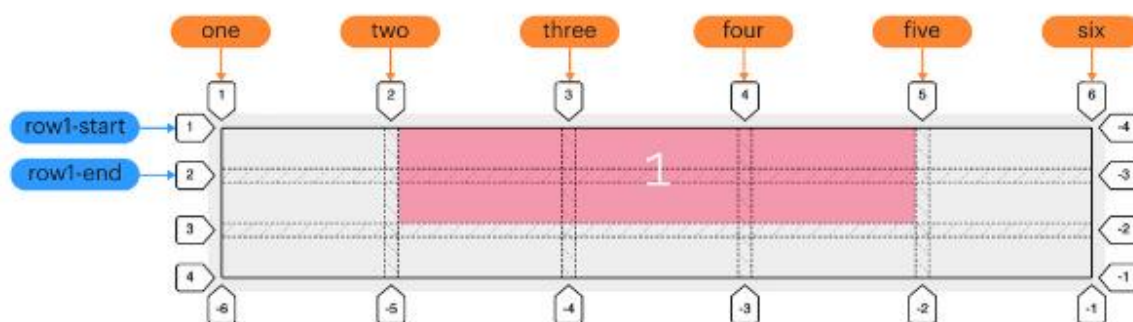


Рисунок 17.3.1 - Свойства `grid-column-start`, `grid-column-end`, `grid-row-start`, `grid-row-end`

Элемент разместился по горизонтали от второй грид-линии до линии с названием `[five]`, а по вертикали — от линии с именем `[row1-start]` до линии под номером 3.

### 17.3.2 Свойства `grid-column` и `grid-row`

Свойства-шорткаты для `grid-column-start`, `grid-column-end` и `grid-row-start`, `grid-row-end`. Значения для `*-start` и `*-end` разделяются слэшем.

Можно использовать ключевое слово `span`, буквально говорящее «*растянуться на столько-то*». А на сколько именно указывает стоящая за ним цифра (листинг 17.3.2 и рисунок 17.3.2).

Листинг 17.3.2 - Свойства `grid-column` и `grid-row`

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

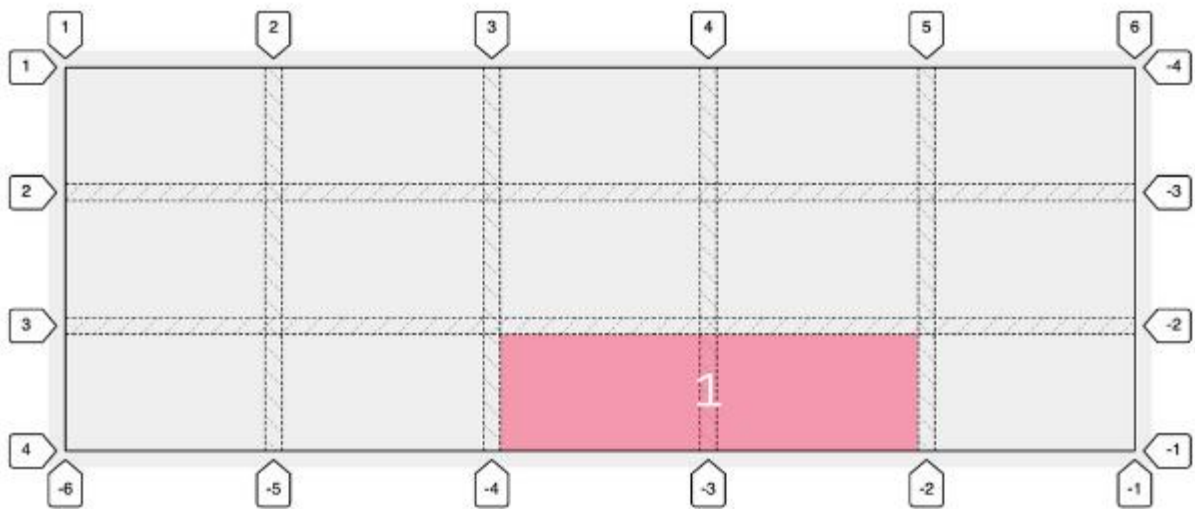


Рисунок 17.3.2 - Свойства `grid-column` и `grid-row`

Элемент начинается с третьей линии по горизонтали и растягивается на 2 ячейки. По вертикали элемент начинается от третьей линии и заканчивается у четвёртой линии.

Если опустить слэш и второе значение, то элемент будет размером в одну ячейку.

### 17.3.3 Свойство `grid-area`

Свойство, которое может указывать элементу, какую из именованных областей ему нужно занять (листинг 17.3.3), а также служить шорткатом для одновременного указания значений для четырёх свойств: `grid-row-start`, `grid-column-start`, `grid-row-end`, `grid-column-end` – именно в таком порядке, т.е. сначала указываем оба начала, после – оба конца (листинг 17.3.4).

Пример с указанием на именованную область (листинг 17.3.3):

Листинг 17.3.3 - Свойства `grid-area` с именованной областью

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

Листинг 17.3.4 - Свойства `grid-area` с четырьмя свойствами

```
.item2 {  
  /* row-start / column-start / row-end / column-end */  
  grid-area: 1 / col4-start / last-line / 6;  
}
```

### 17.3.4 Свойство `justify-self`

С помощью свойства `justify-self` можно установить горизонтальное выравнивание для отдельного элемента, отличное от выравнивания, заданного грид-родителю (листинг 17.3.5, рисунок 17.3.3). Возможные значения аналогичны значениям свойства `justify-items`.

Листинг 17.3.5 - Свойство `justify-self`

```
.container {  
  justify-items: stretch;  
}  
  
.item1 {  
  justify-self: center;  
}
```

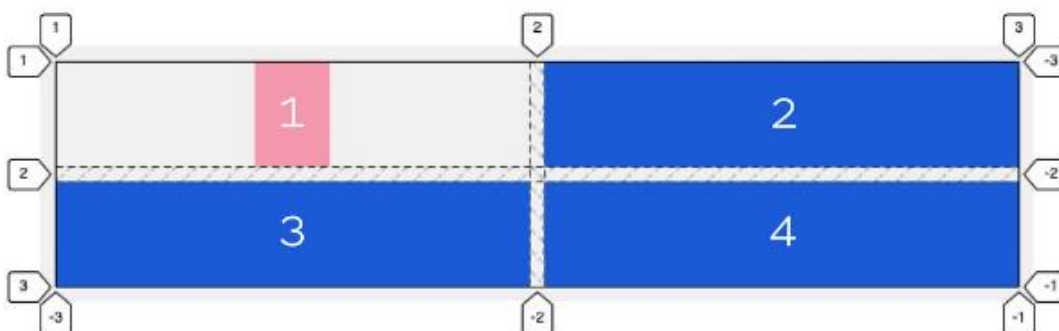


Рисунок 17.3.3 - Свойство `justify-self`

### 17.3.5 Свойство align-self

Свойство `align-self` выравнивает отдельный элемент по вертикальной оси. Возможные значения аналогичны значениям свойства `align-items` (личтинг 17.3.6, рисунок 17.3.4).

### Листинг 17.3.6 - Свойство align-self

```
.container {
    align-items: stretch;
}

.item1 {
    align-self: start;
}
```

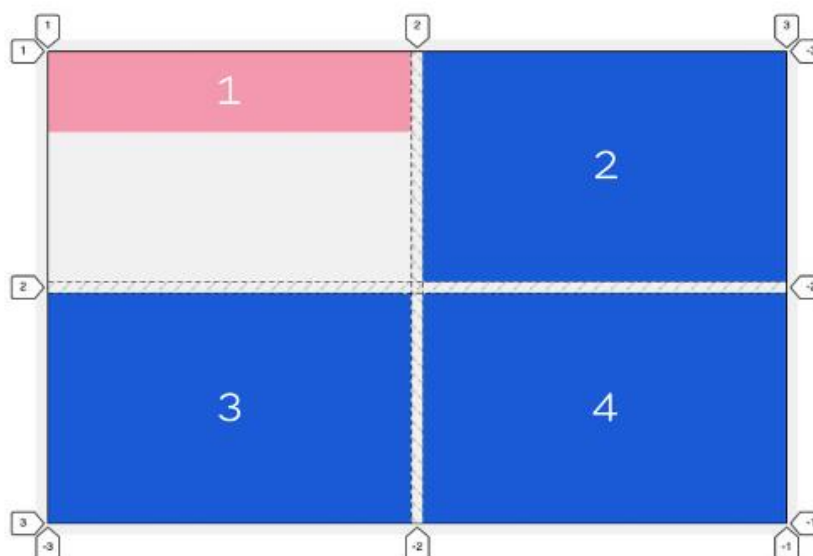


Рисунок 17.3.4 - Свойство align-self

### 17.3.6 Свойство place-self

Свойство-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.

### Дополнительные функции и ключевые слова

Когда вы задаёте размеры колонкам и рядам, вам доступны не только известные единицы измерения (*px*, *vw*, *rem*, *%* и так далее), но и ключевые слова `min-content`, `max-content`, `fit-content` и `auto`. И уже упомянутые единицы измерения `fr`.

Гриды подарили нам ещё одну функцию, позволяющую одновременно задавать минимальный и максимальный размер — `minmax()`. Например, в случае записи `grid-template-columns: minmax(200px, 1fr)`; колонка займёт 1 часть свободного пространства грид-контейнера, но не меньше 200 пикселей.

Подробнее про ключевые слова `min-content`, `max-content`, `fit-content` и `auto` вы можете прочитать в статье: <https://webformyself.com/min-content-max-content-i-fit-content-v-css/> (русс. яз.).

### Анимация свойств

Для анимации доступны следующие свойства и их значения:

- Значения свойств `gap`, `row-gap`, `column-gap`, указанные в единицах измерения, процентах или при помощи `calc()`.
- Значения свойств `grid-template-columns`, `grid-template-rows`, указанные в единицах измерения, процентах или при помощи функции `calc()` при условии, что анимируются аналогичные значения.

### Когда нужно использовать flexbox, а когда grid?

Гриды являются лучшим выбором, когда у вас есть **сложные проекты, требующие вёрстки с использованием двумерной компоновки**. Эта система позволяет создавать вёрстку со сложным дизайном, который легко обслуживается и требует меньше усилий при поддержке и масштабировании. Кроме того, свойство `gap` упрощает определение отступов между элементами и использование его для создания перекрывающихся элементов. **Если вы работаете над проектом впервые и хотите создавать структуру и дизайн будущей вёрстки с помощью гридов, то эта система будет очень полезной.**

Во всех остальных случаях можно смело использовать флексбоксы.



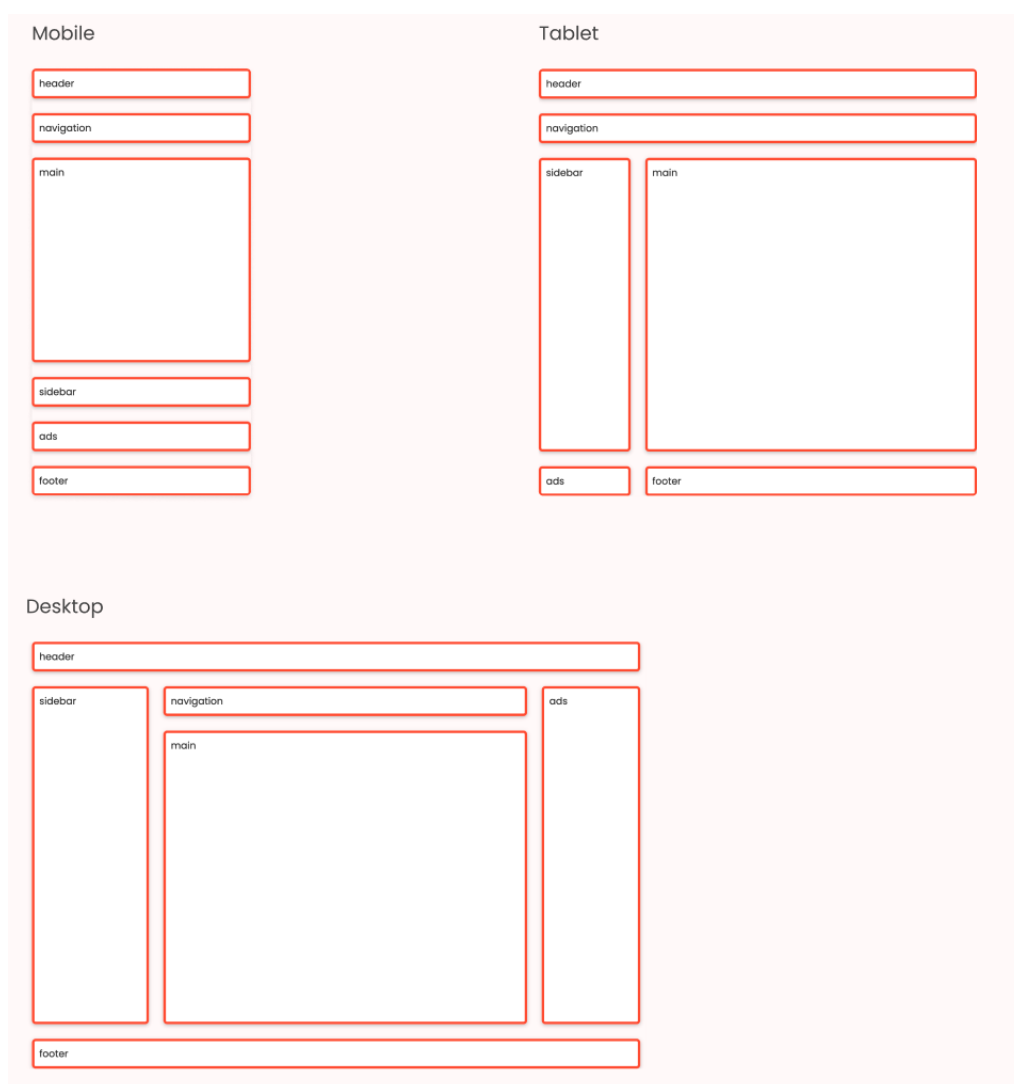
### Дополнительная информация

1. A Complete Guide to Grid – URL: <https://css-tricks.com/snippets/css/complete-guide-grid/> (англ. яз.)
2. Grid Cheatsheet – URL: <https://yoksel.github.io/grid-cheatsheet/> (англ.яз.)
3. Игра GRID GARDEN – URL: <https://cssgridgarden.com/#ru> (русс. яз.)
4. GRID cheat sheet (наглядное отображение) – URL: <https://grid.malven.co/> (англ.яз.)

## Практическое задание

**Примечание.** В каждом элементе должны быть использованы CSS-переменные и адаптивность. Контент страницы всегда должен пропорционально уместаться на экране устройств с шириной от 320px до 2560px.

1. Реализовать адаптивность своего сайта по семантическим элементам под различные устройства с помощью Grid Layout. Пример адаптивного макета отображения вашего сайта с помощью Grid Layout представлен на рисунке ниже (**можно использовать макет своего собственного сайта – это приветствуется**):



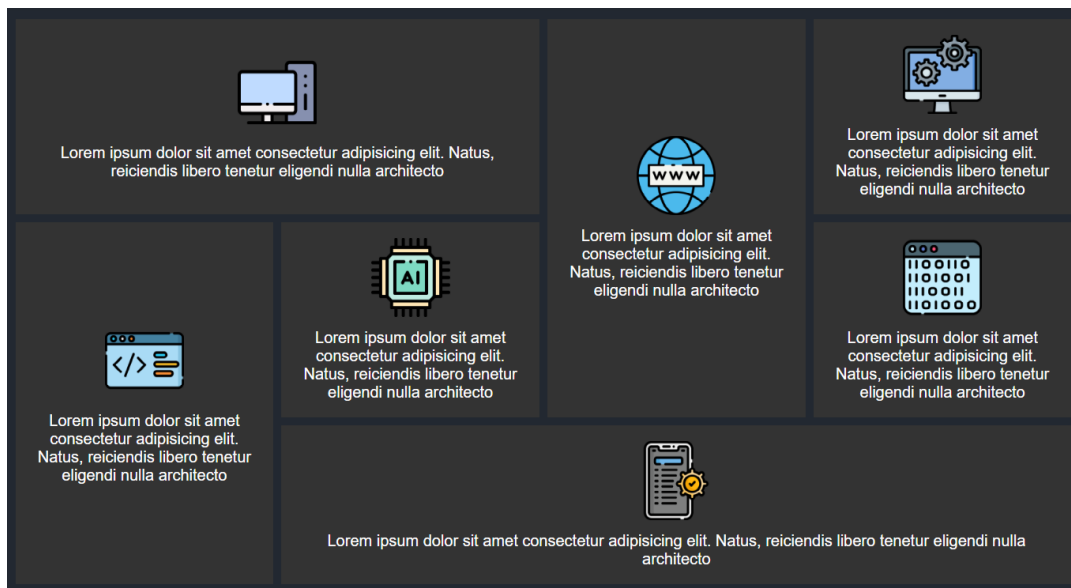
2. При выполнении задания выбрать один из двух предложенных ниже вариантов.

**Вариант №1**

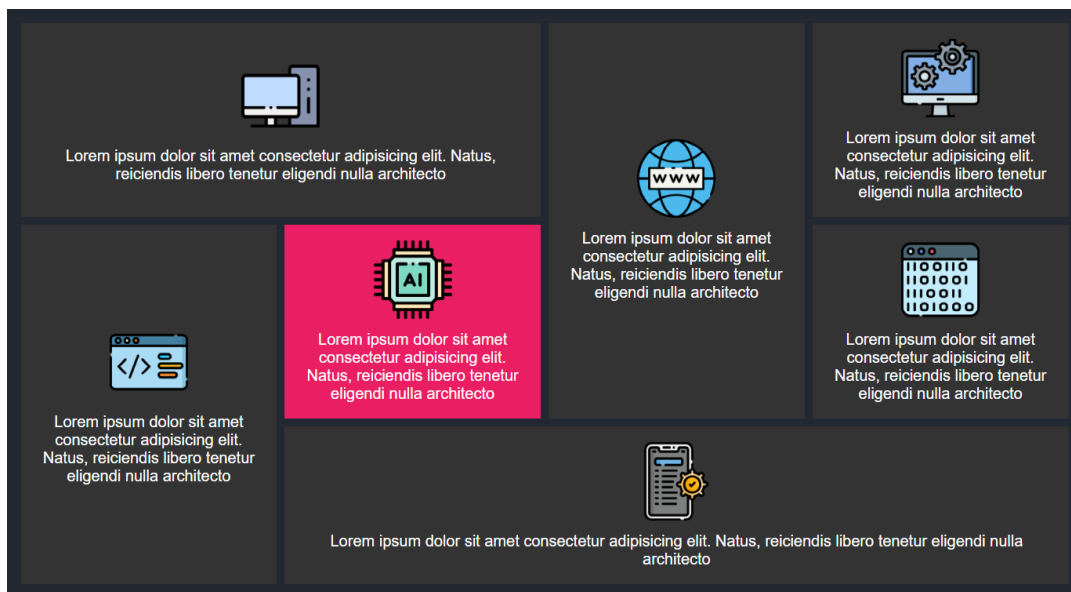
Создать сетку из 7 блоков при помощи свойства модуля Grid Layout, [transition](#), [transform](#). Добавить плавную анимацию при наведении.

Пример:

До наведения:



После наведения:

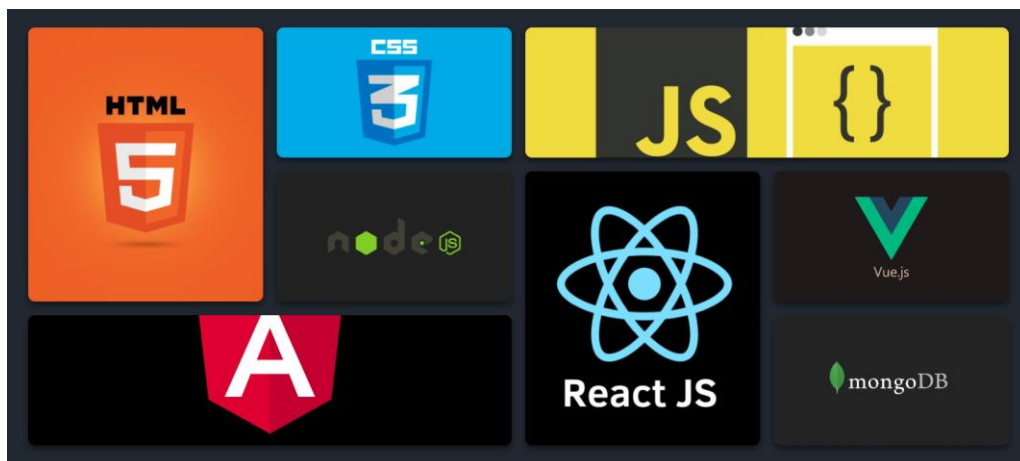


## Вариант №2

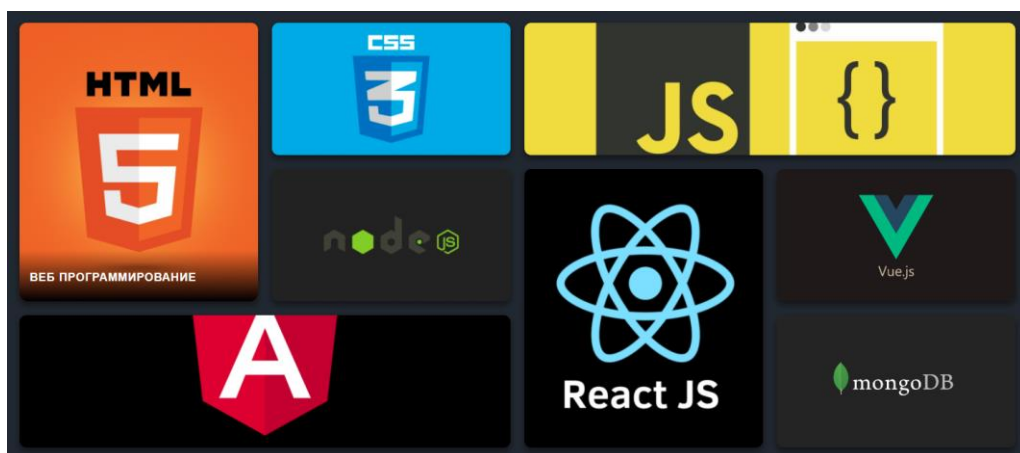
Перестроить созданную ранее галерею под мозаичный вид при помощи свойства модуля Grid Layout, [transition](#), [transform](#). Добавить плавную анимацию при наведении.

Пример:

До наведения:



После наведения:



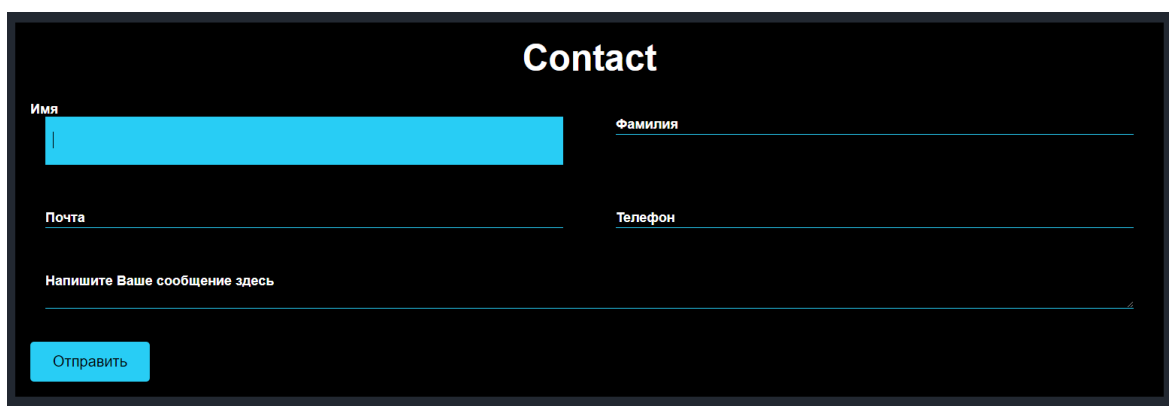
3. Создать форму (или изменить ранее созданную). Использовать свойства модуля Grid Layout, `transition`, `transform`. Добавить плавную анимацию при наведении и клике (псевдокласс `:focus`) и придумать свой дизайн формы.

Пример:

До наведения/клика:

A contact form titled 'Contact' with a dark background. It features five input fields: 'Имя' (Name), 'Фамилия' (Surname), 'Почта' (Email), 'Телефон' (Phone), and a larger text area for a message with the placeholder 'Напишите Ваше сообщение здесь'. A blue 'Отправить' (Send) button is located at the bottom left.

После наведения/клика:



**Contact**

Имя

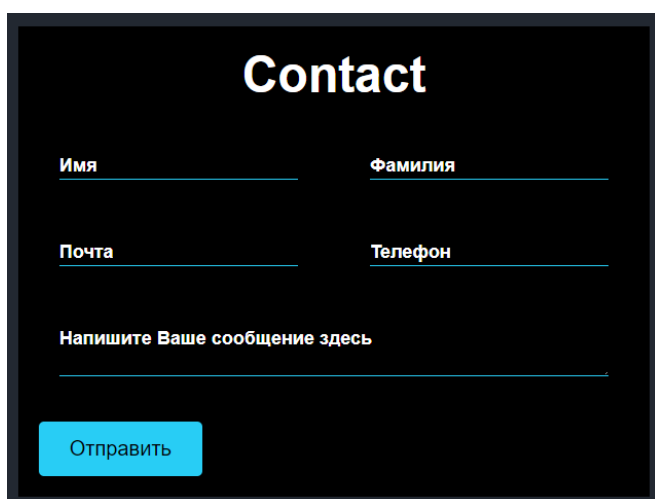
Фамилия

Почта

Телефон

Напишите Ваше сообщение здесь

Адаптив (713-2560px):



**Contact**

Имя

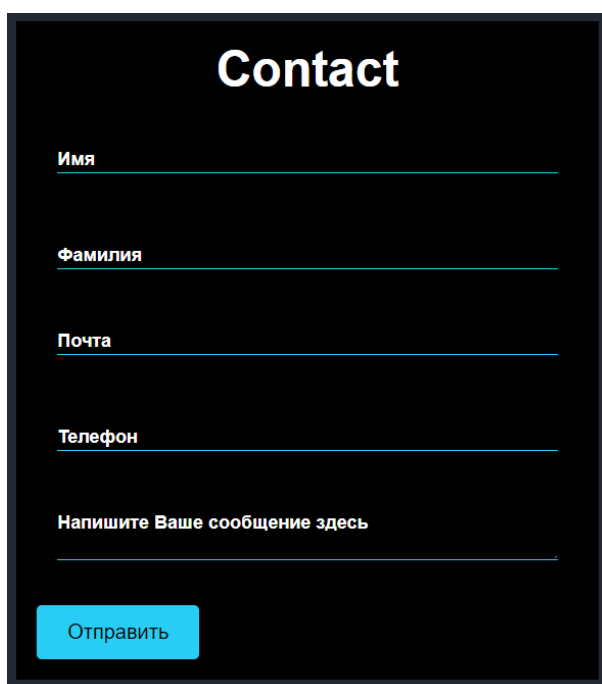
Фамилия

Почта

Телефон

Напишите Ваше сообщение здесь

Адаптив (320-712px):



**Contact**

Имя

Фамилия

Почта

Телефон

Напишите Ваше сообщение здесь

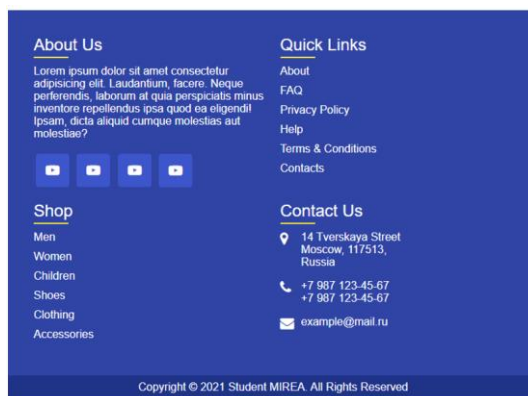
4. В футере использовать свойства модуля Grid Layout, [transition](#), [transform](#) и псевдоэлементы. Добавить плавную анимацию при наведении.

Пример:

Адаптив (1001-2560px):



Адаптив (769-1000px):



Адаптив (320-768px):

