**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.

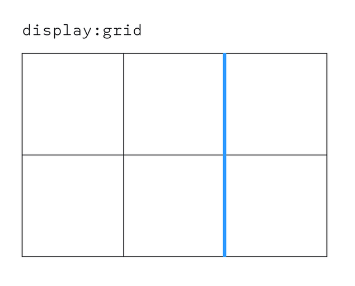


Рисунок 17.1 - Grid-линия колонки

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

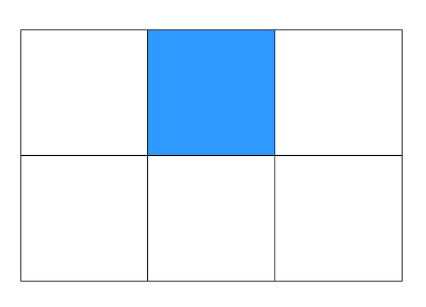


Рисунок 17.2 - Grid-ячейка между первой и второй grid-линиями ряда и второй и третьей grid-линиями колонки

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

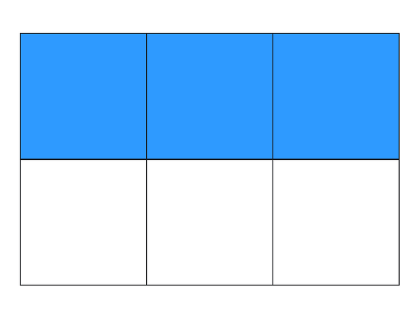


Рисунок 17.3 - Горизонтальная grid-полоса между первой и второй grid-линиями ряда

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

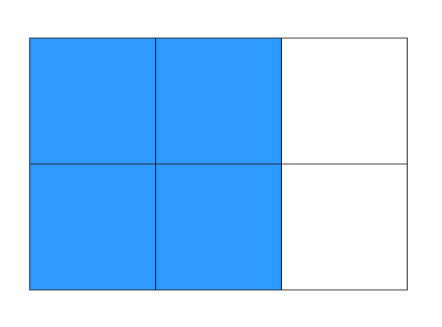


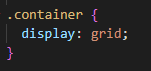
Рисунок 17.4 - Grid-область между первой и третьей grid-линиями ряда и первой и третьей grid-линиями колонки

## 17.2 Свойства grid-контейнера

## 17.2.1 Свойство display

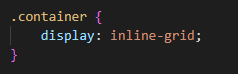
Если элементу задано свойство display со значением grid, то такой элемент становится грид-контейнером (листинг 17.2.1). Дочерние элементы этого контейнера начинают подчиняться правилам грид-раскладки. Снаружи грид-контейнер ведёт себя как блок.

Листинг 17.2.1 – Значение grid



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).

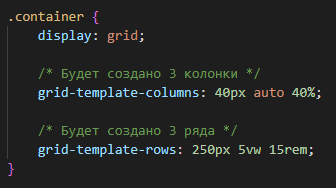
Листинг 17.2.2 - Значение inline-grid



## 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



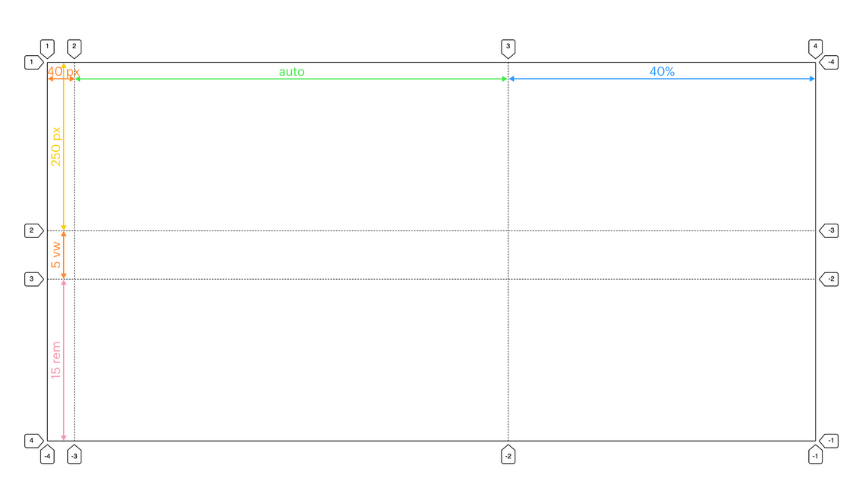
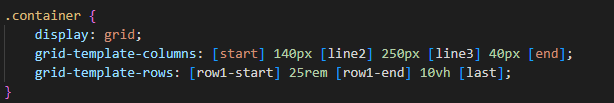


Рисунок 17.2.1 - Свойства grid-template-columns и grid-template-rows

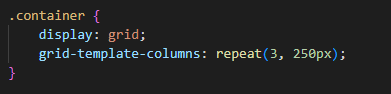
Вы можете также явно именовать грид-линии, используя для этого квадратные скобки (листинг 17.2.4). Каждая линия также может иметь больше одного имени – они пишутся через пробел.

Листинг 17.2.4 – Именование 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):

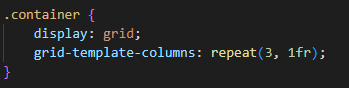
Листинг 17.2.5 – Именование grid-линий



С появлением гридов появилась новая единица измерения: fr.

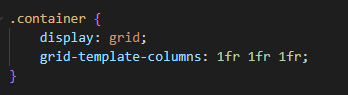
fr (от *fraction* — доля, часть) отвечает за свободное пространство внутри грид-контейнера. Например, код ниже создаст три колонки, каждая из которых будет занимать 1/3 ширины родителя (листинг 17.2.6):

Листинг 17.2.6 – Использование долей (fr) – сокращенное написание



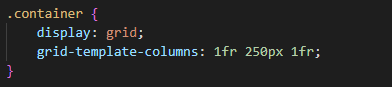
Верхняя запись кода аналогична нижепредставленной (листинг 17.2.7):

Листинг 17.2.7 – Использование долей (fr) – полное написание



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% - 250px) / 2 (листинг 17.2.8):

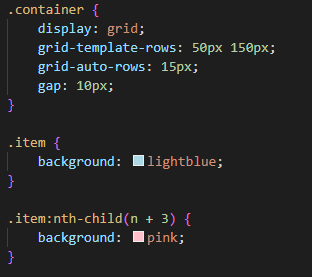
Листинг 17.2.8 – Распределение свободного пространства



## 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 17.2.2).

Листинг 17.2.9 – Свойство grid-auto-rows



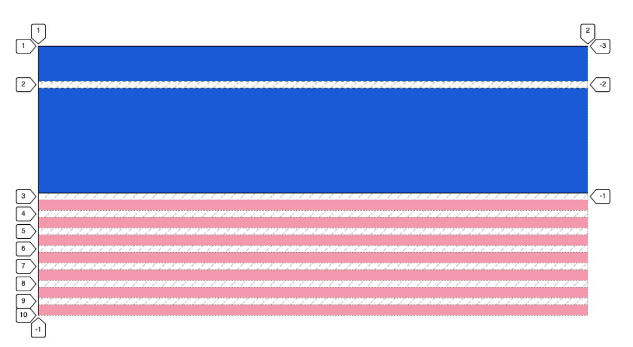


Рисунок 17.2.2 – Свойство grid-auto-rows

В этом примере создаются два явных ряда размером 50 и 150 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 15 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 17.2.3 и листинг 17.2.10).

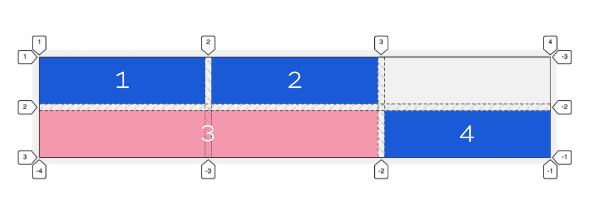
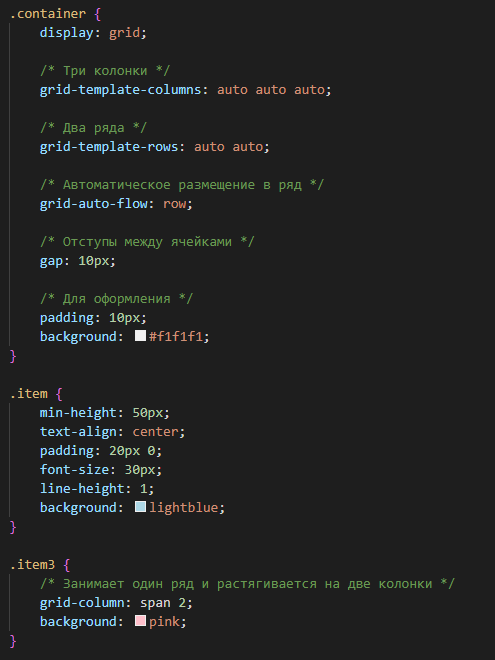


Рисунок 17.2.3 – Свойство grid-auto-flow

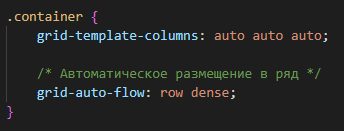
Как видно на рисунке 17.2.3, третий элемент не поместился в последнюю ячейку первого ряда и был перенесён в следующий ряд. Следующий за ним четвёртый элемент встал в ближайшую доступную ячейку во втором ряду.

Листинг 17.2.10 – Свойство grid-auto-flow



Исправить данную ситуацию можно с помощью добавления значения dense к свойству grid-auto-flow (листинг 17.2.11 и рисунок 17.2.4):

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



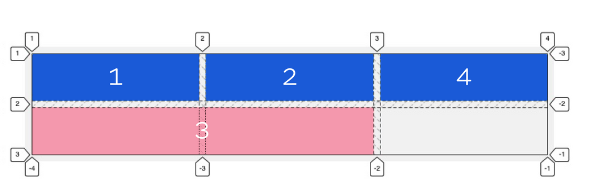


Рисунок 17.2.4 – Свойство grid-auto-flow со значением dense

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.

## 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.

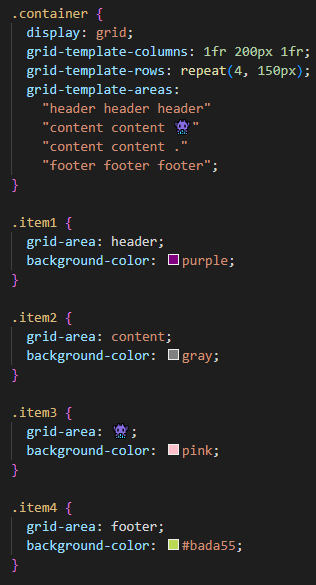
Возможные значения:

* 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 17.2.5).

**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



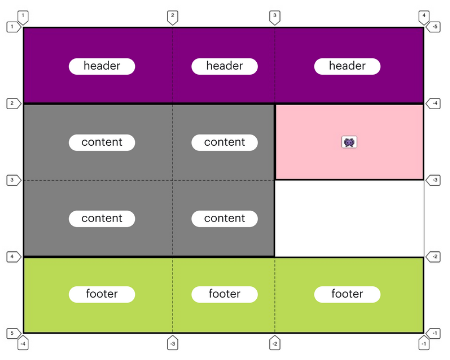


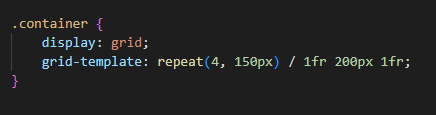
Рисунок 17.2.5 - Свойство grid-template-areas

## 17.2.6 Свойство grid-template

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

Можно прописать все колонки и ряды сразу, разделяя их слэшем /. **Сперва идут ряды, а затем колонки** (листинг 17.2.13).

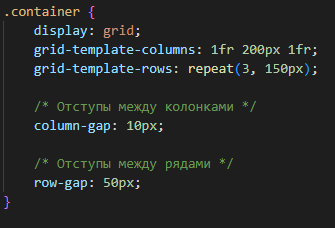
Листинг 17.2.13 – Свойство grid-template

****

## 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 17.2.6).

Листинг 17.2.14 – Свойства column-gap и row-gap



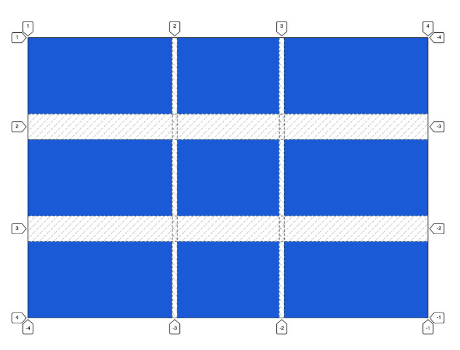
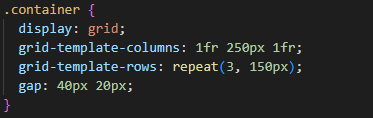


Рисунок 17.2.5 - Свойства column-gap и row-gap

## 17.2.8 Свойство gap

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

Листинг 17.2.15 – Свойство gap



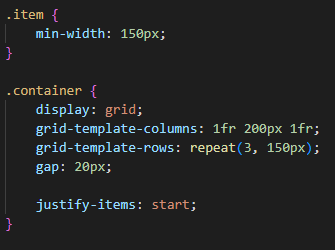
## 17.2.9 Свойство justify-items

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

Возможные значения:

* start - выравнивает элемент по начальной линии (листинг 17.2.16 и рисунок 17.2.6).

Листинг 17.2.16 – Свойство justify-items со значением start



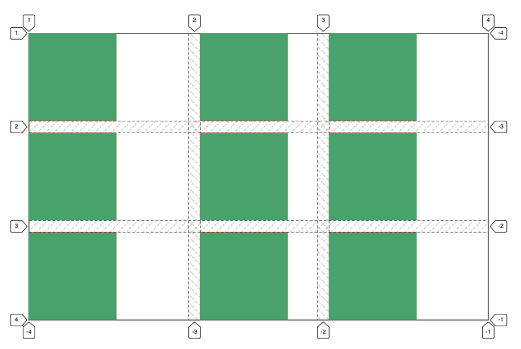
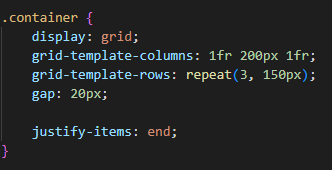


Рисунок 17.2.6 - Свойство justify-items со значением start

* end - выравнивает элемент по конечной линии (листинг 17.2.17 и рисунок 17.2.7).

Листинг 17.2.17 – Свойство justify-items со значением end



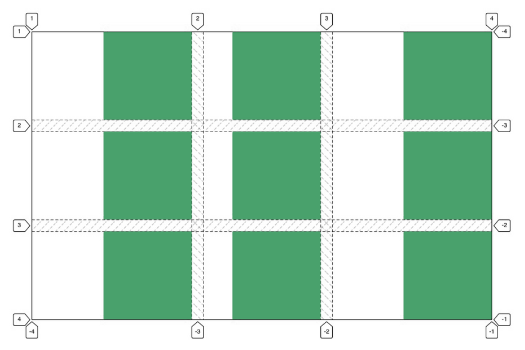
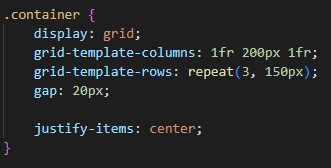


Рисунок 17.2.7 - Свойство justify-items со значением end

* center - выравнивает элемент по центру грид-ячейки (листинг 17.2.18 и рисунок 17.2.8).

Листинг 17.2.18 – Свойство justify-items со значением center



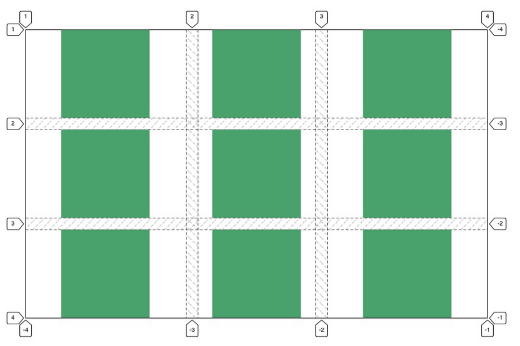
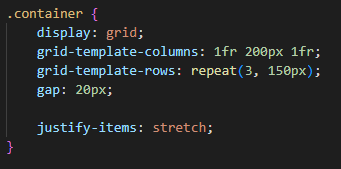


Рисунок 17.2.8 - Свойство justify-items со значением center

* stretch - растягивает элемент на всю ширину грид-ячейки (листинг 17.2.19 и рисунок 17.2.9).

Листинг 17.2.19 – Свойство justify-items со значением stretch



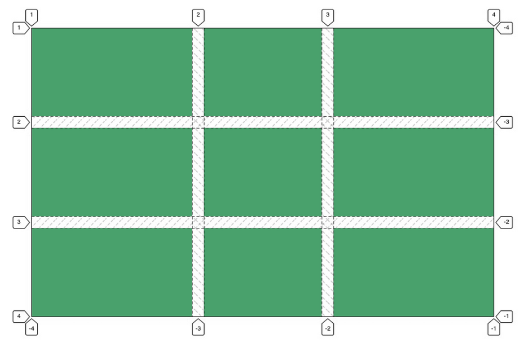


Рисунок 17.2.9 - Свойство justify-items со значением stretch

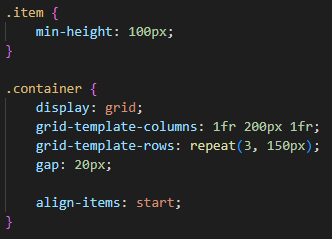
## 17.2.10 Свойство align-items

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 17.2.10).

Листинг 17.2.20 – Свойство align-items со значением start



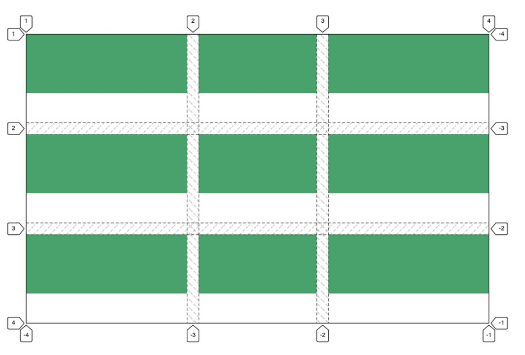


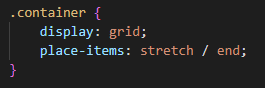
Рисунок 17.2.10 - Свойство align-items со значением start

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

## 17.2.11 Свойство place-items

Свойство-шорткат для указания значений сразу и для align-items и для justify-items (листинг 17.2.21). Порядок указания свойств должен быть именно таким.

Листинг 17.2.21 – Свойство place-items



## 17.2.12 Свойство grid

Свойство, позволяющее задать значения всем свойствам сразу. А именно для:

* 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 этот свойство.
* 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, если слева, то - ряды.
* 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.

## 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.

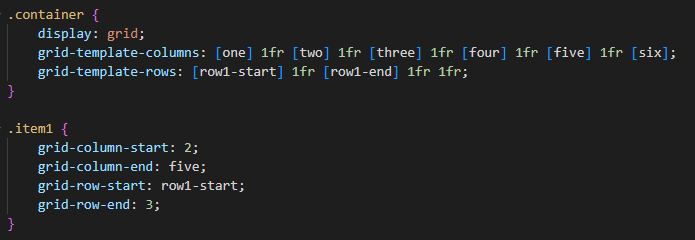
## 17.3.1 Свойства grid-column-start, grid-column-end, grid-row-start, grid-row-end

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



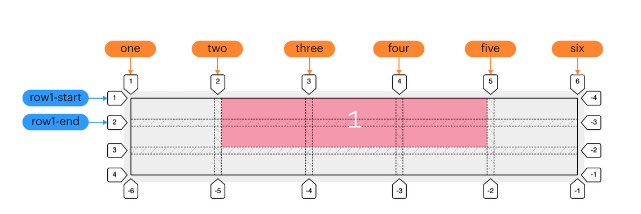


Рисунок 17.3.1 - Свойства grid-column-start, grid-column-end, grid-row-start, grid-row-end

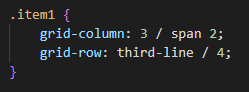
Элемент разместился по горизонтали от второй грид-линии до линии с названием [five], а по вертикали — от линии с именем [row1-start] до линии под номером 3.

## 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



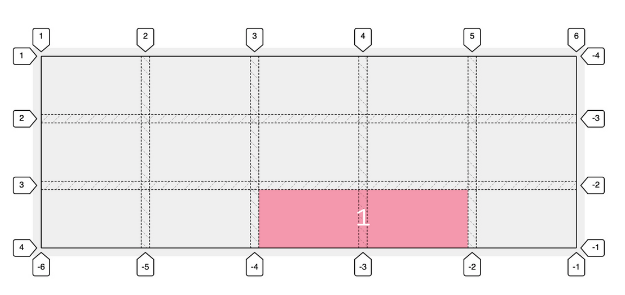


Рисунок 17.3.2 - Свойства grid-column и grid-row

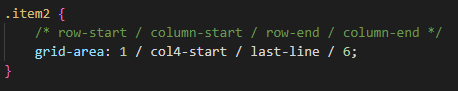
Элемент начинается с третьей линии по горизонтали и растягивается на 2 ячейки. По вертикали элемент начинается от линии с именем [third-line] и заканчивается у четвёртой линии.

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

## 17.3.3 Свойство grid-area

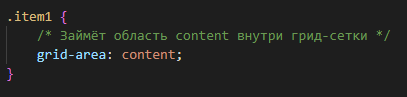
Свойство, которое может указывать элементу, какую из именованных областей ему нужно занять, а также служить шорткатом для одновременного указания значений для четырёх свойств: grid-row-start, grid-column-start, grid-row-end, grid-column-end – именно в таком порядке, т.е. сначала указываем оба начала,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

Листинг 17.3.3 - Свойства grid-area с четырьмя свойствами



Пример с указанием на именованную область (листинг 17.3.4):

Листинг 17.3.4 - Свойства grid-area с именованной областью



## 17.3.4 Свойство justify-self

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

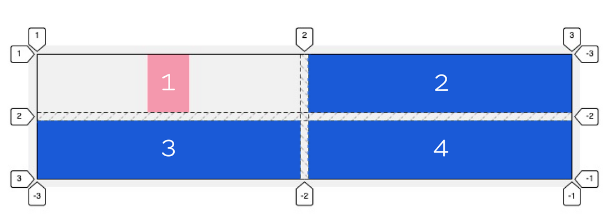


Рисунок 17.3.3 - Свойство justify-self

## 17.3.5 Свойство align-self

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

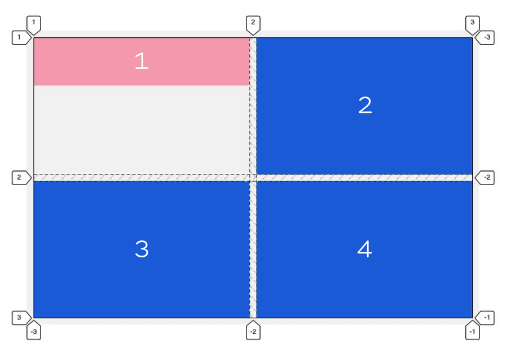


Рисунок 17.3.4 - Свойство align-self

## 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.).

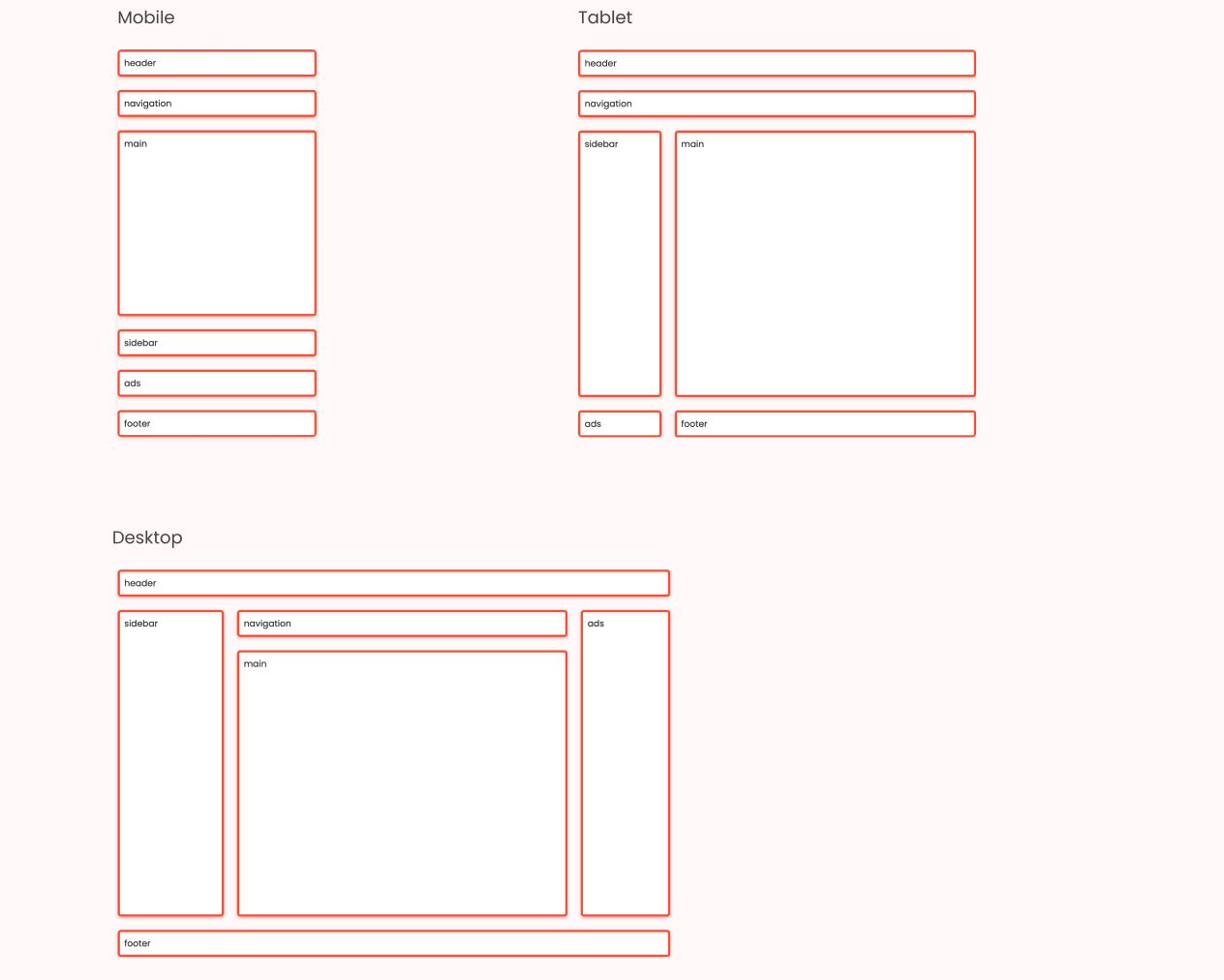
## Дополнительная информация

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 умещаться 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:



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

**Вариант №1**

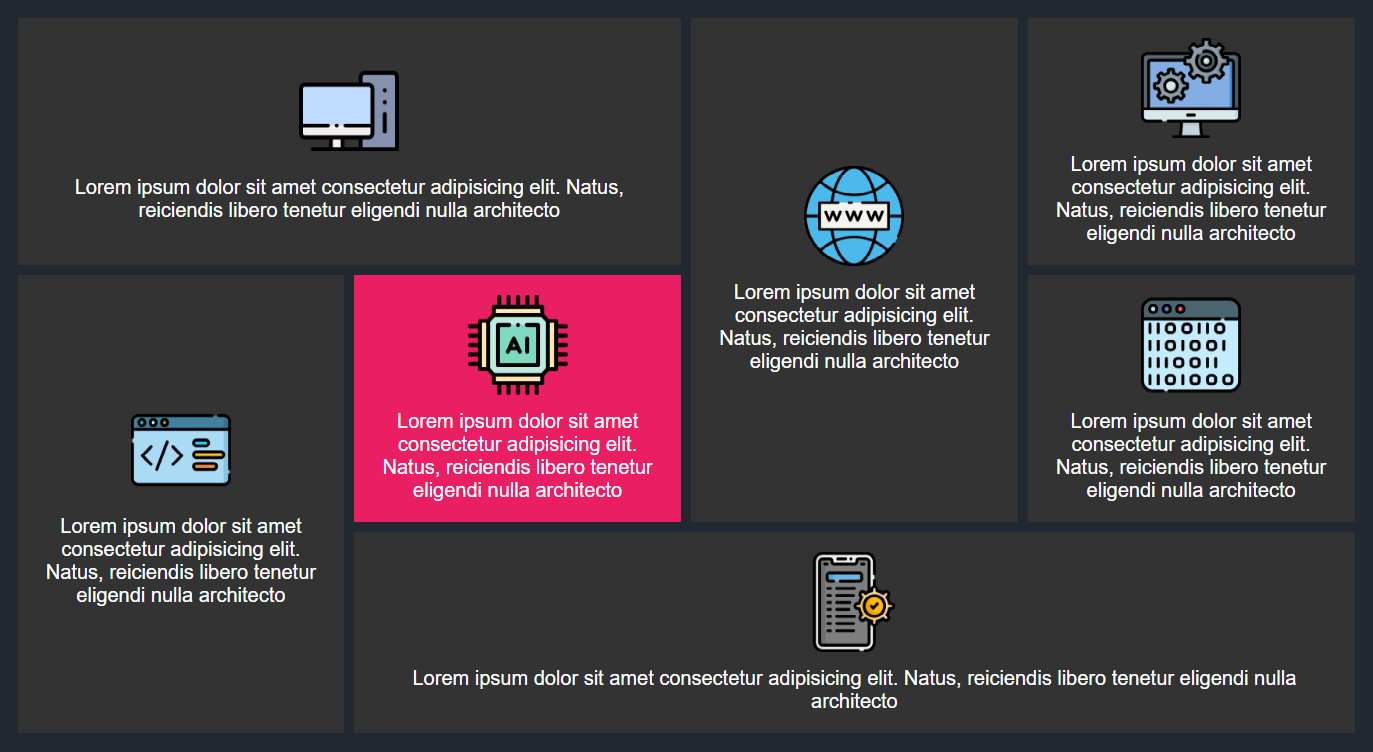
Создать сетку из 7 блоков при помощи свойства модуля Grid Layout, transition, transform. Добавить плавную анимацию при наведении.

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:



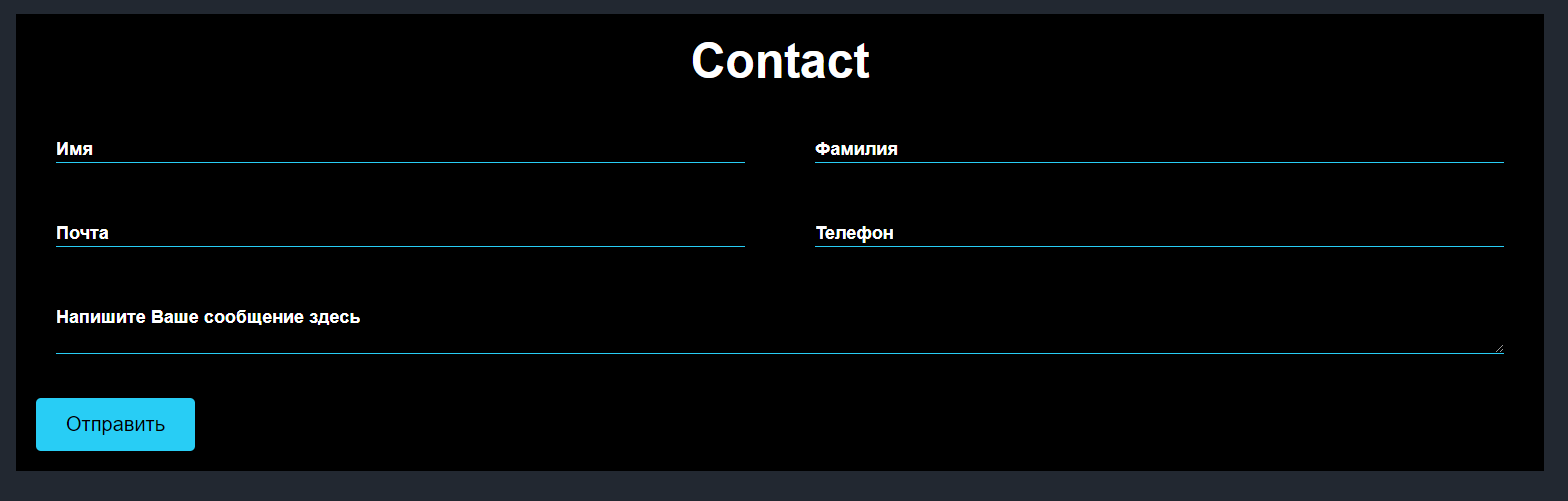
После наведения:



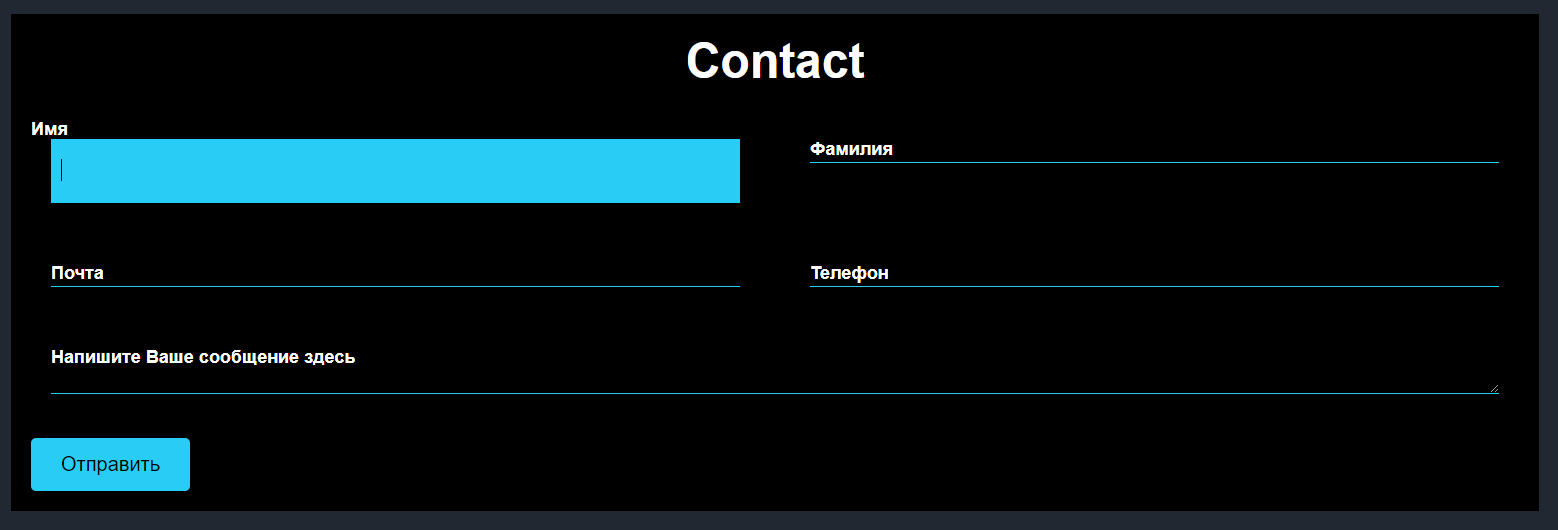
**3.** Создать форму (или изменить ранее созданную). Использовать свойства модуля Grid Layout, transition, transform. Добавить плавную анимацию при наведении и клике (псевдокласс :focus) и придумать свой дизайн формы.

Пример:

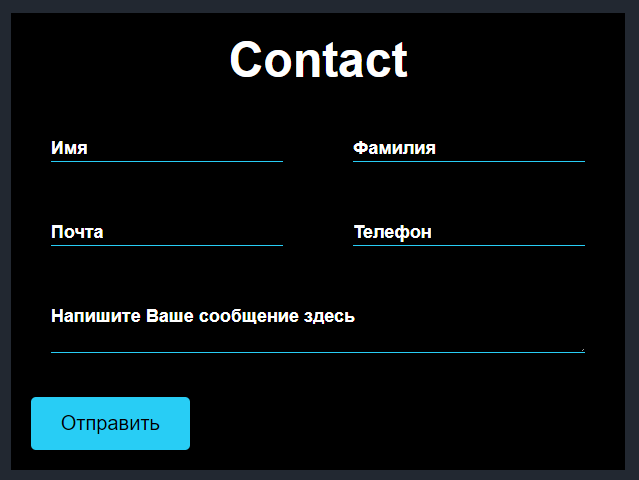
До наведения/клика:



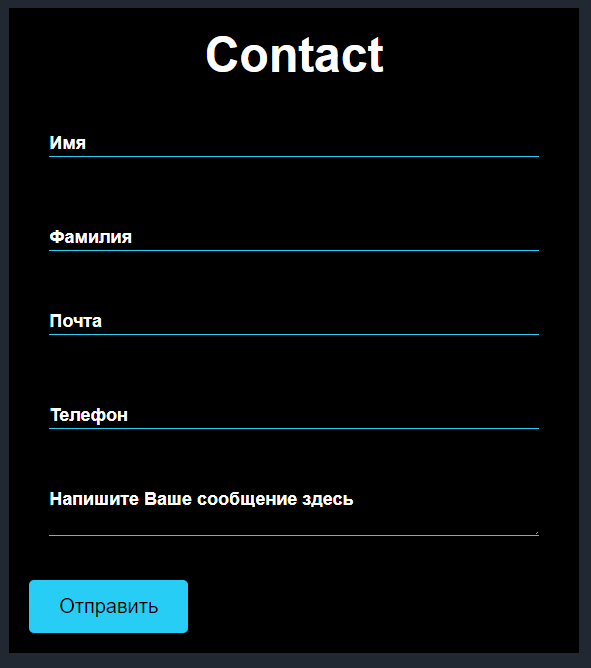
После наведения/клика:



Адаптив (713-2560px):



Адаптив (320-712px):



**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):

