



# CSS

Cascading Style  
Sheets  
(Каскадные  
таблицы стилей.)

# ПЛАН



1

SVG, css import.

2

Flexbox.

3

Media Queries.

4

Transform.

# ИКОНКИ

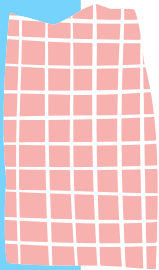
Иконки CSS - это графические элементы, которые можно создавать и стилизовать с помощью CSS, вместо использования отдельных изображений. Использование иконок CSS имеет ряд преимуществ, включая гибкость, масштабируемость и возможность изменять иконки динамически с помощью CSS свойств.



- Векторные графики представляют собой графические изображения, созданные с использованием математических формул, что позволяет им быть масштабируемыми без потери качества.
- SVG (Scalable Vector Graphics) - это формат векторной графики, который можно встраивать в HTML и стилизовать с помощью CSS.
- Возможности стилизации иконок CSS могут включать изменение размеров, цвета, тени, фона, анимации и многое другое, в зависимости от того, какие свойства CSS вы применяете к элементам с классами иконок.

Использование иконок CSS позволяет создавать динамические, гибкие и легко настраиваемые графические элементы на веб-страницах. Они являются альтернативой традиционным изображениям и позволяют легко изменять внешний вид и поведение иконок с помощью CSS, без необходимости создания или загрузки дополнительных изображений. Это экономит пропускную способность, повышает производительность и облегчает обслуживание веб-сайта.

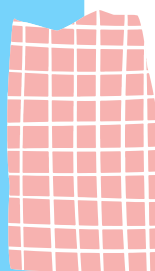
# IMPORT



Синтаксис @import позволяет вам импортировать один CSS-файл в другой. Вы можете разместить инструкцию @import в любом месте CSS-файла, где вы хотите импортировать другой файл.

```
@import 'text.css';
```

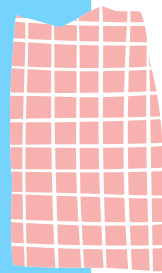
# **FLEXBOX**



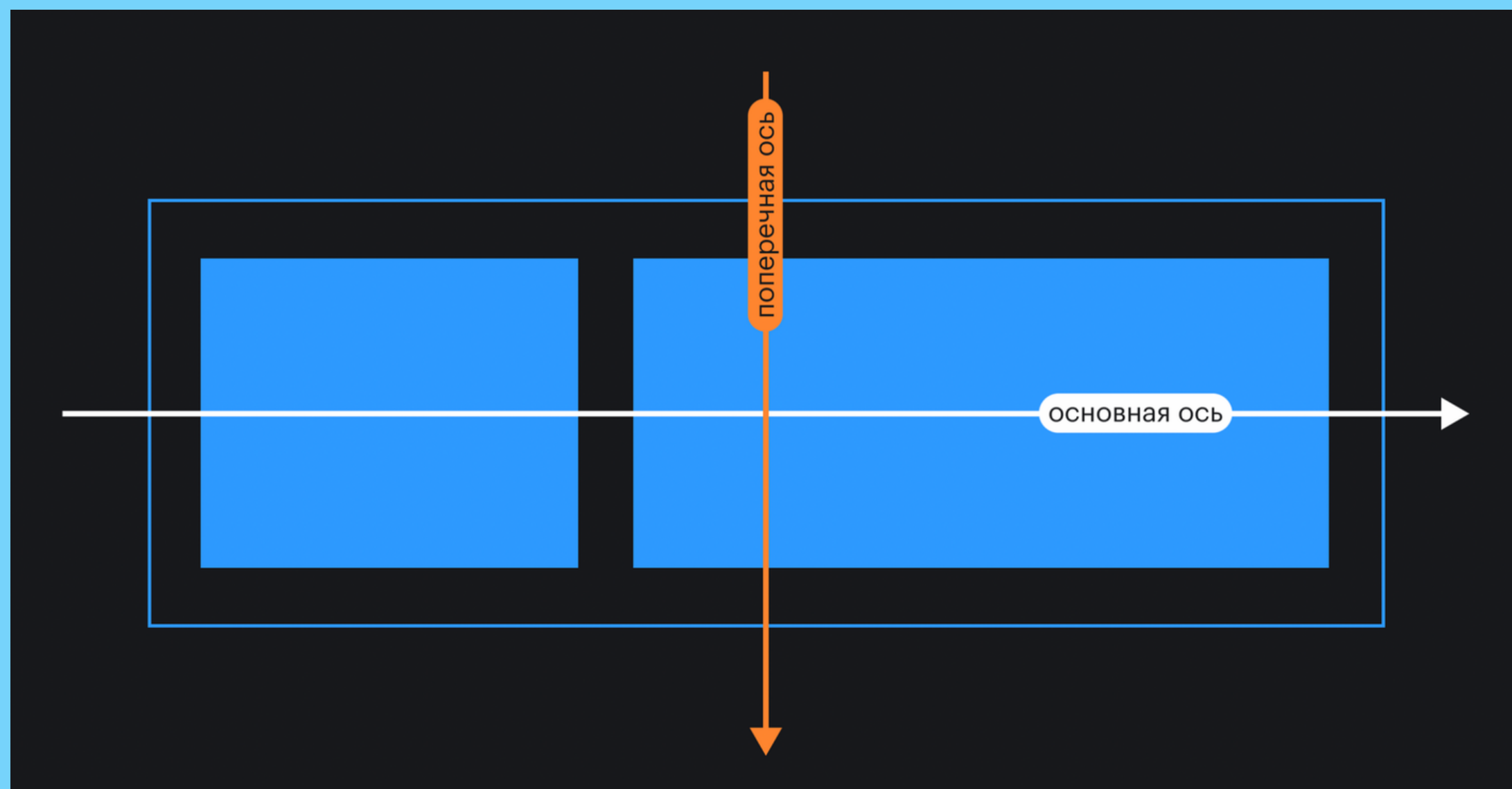
Идея флексбоксов появилась ещё в 2009 году, и этот стандарт до сих пор развивается и прорабатывается. Основная идея флексов — гибкое распределение места между элементами, гибкая расстановка, выравнивание, гибкое управление. Ключевое слово — гибкое, что и отражено в названии (flex — англ. гибко).

# СВОЙСТВА ФЛЕКС-КОНТЕЙНЕРА

- Когда мы задаём какому-то элементу значение flex для свойства display, мы превращаем этот элемент в флекс-контейнер. Внутри него начинает действовать флекс-контекст, его дочерние элементы начинают подчиняться свойствам флексбокса.
- Снаружи флекс-контейнер выглядит как блочный элемент — занимает всю ширину родителя, следующие за ним элементы в разметке переносятся на новую строку.



- Основная ось: основная направляющая флекс-контейнера, вдоль которой располагаются флекс-элементы.
- Поперечная (побочная, перпендикулярная) ось: ось, идущая перпендикулярно основной.





- Флекс-контейнер: элемент, к которому применяется свойство `display: flex`. Вложенные в него элементы подчиняются правилам раскладки флексов.
- Флекс-элемент: элемент, вложенный во флекс-контейнер.



#### Свойства контейнера Flex:

- `flex-direction`
- `flex-wrap`
- `flex-flow`
- `justify-content`
- `align-items`
- `align-content`

## flex-direction

1. Основная ось (Main Axis): Определяется свойством flex-direction, которое может быть установлено на row (горизонтально), column (вертикально), row-reverse (горизонтально в обратном порядке) или column-reverse (вертикально в обратном порядке).

## flex-wrap

1. По умолчанию значение у свойства flex-wrap — nowrap. При этом флекс-элементы помещаются (или пытаются уместиться) в один ряд и не переносятся в новый ряд, даже если не влезают в размеры родителя.
2. Установив значение wrap, мы можем изменить это поведение, и флекс-элементы будут иметь возможность перенестись в новый ряд, если не влезают в одну линию в рамках родителя.
3. Ещё одно возможное значение — wrap-reverse. В этом случае элементы будут располагаться снизу вверх, заполнив собой сперва нижний ряд, а те, что не влезли, перепрыгнут в ряд выше.

`flex-flow`

Это свойство-шорткат для одновременного определения значений свойств `flex-direction` и `flex-wrap`.

## justify-content

Выравнивает элементы по основной оси: если `direction=row`, то по горизонтали, а если `direction=column`, то по вертикали.

- `flex-start (default)` - элементы будут идти с начала (в конце может остаться место).
- `flex-end` - элементы выравниваются по концу (место останется в начале)
- `center` - по центру (место останется слева и права)
- `space-between` - крайние элементы прижимаются к краям (место между элементами распределяется равномерно)
- `space-around` — свободное пространство делится поровну между элементами и по половине от этой доли размещается по бокам от каждого элемента. Таким образом, между соседними элементами будет равное расстояние, а снаружи крайних элементов — по половине этого расстояния.
- `space-evenly` — свободное место будет распределено так, чтобы расстояние между любыми двумя элементами было одинаковым и расстояние от крайних элементов до края было таким же.

## align-items

Свойство align-items выравнивает флекс-элементы внутри контейнера в перпендикулярном направлении.

У свойства есть пять возможных значений:

- flex-start: элементы выравниваются по верхнему краю контейнера;
- flex-end: элементы выравниваются по нижнему краю контейнера;
- center: содержимое контейнера выравнивается по центру;
- baseline: элементы выравниваются по базовой линии текста, который в них содержится;
- stretch: внутренние элементы растягиваются на всю высоту flex-контейнера.

## align-content

Свойство распределяет свободное пространство по поперечной оси между рядами флекс-элементов. Предположим, у вас 11 элементов в 3 рядах. Если размер родителя по поперечной оси позволяет, то при помощи align-content можно распределять строки элементов: по верхнему краю, по нижнему, по центру или равномерно.

Не имеет видимого значения, если элементы располагаются в один ряд.

- stretch (значение по умолчанию) — ряды растягиваются одинаково, так, чтобы занять всё доступное пространство родителя.
- flex-start / start — все ряды располагаются у начала поперечной оси. Первое значение не зависит от направления чтения текущего языка, в отличие от второго.
- flex-end / end — все ряды располагаются у конца поперечной оси. end «уважает» направление чтения текущего языка.
- center — ряды выравниваются по центру родителя.
- space-between — первый ряд прижимается к началу поперечной оси, последний — к концу поперечной оси, а остальные располагаются так, чтобы свободное пространство было поделено на отступы между ними равномерно.
- space-around — отступы у каждого ряда равнозначны отступам у любого другого ряда.
- space-evenly — отступы между рядами и от краёв родителя одинаковые.

# СВОЙСТВА ФЛЕКС-ЭЛЕМЕНТА

Свойства элемента flex следующие:

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self



order

При помощи свойства `order` можно менять порядок отображения флекс-элементов внутри флекс-контейнера.

По умолчанию элементы отображаются в том порядке, в котором они расположены в разметке, а значение свойства `order` равно 0.

Значение задаётся в виде целого отрицательного или положительного числа. Элементы встают по возрастающей.

При помощи свойства `order` можно менять порядок отображения флекс-элементов внутри флекс-контейнера.

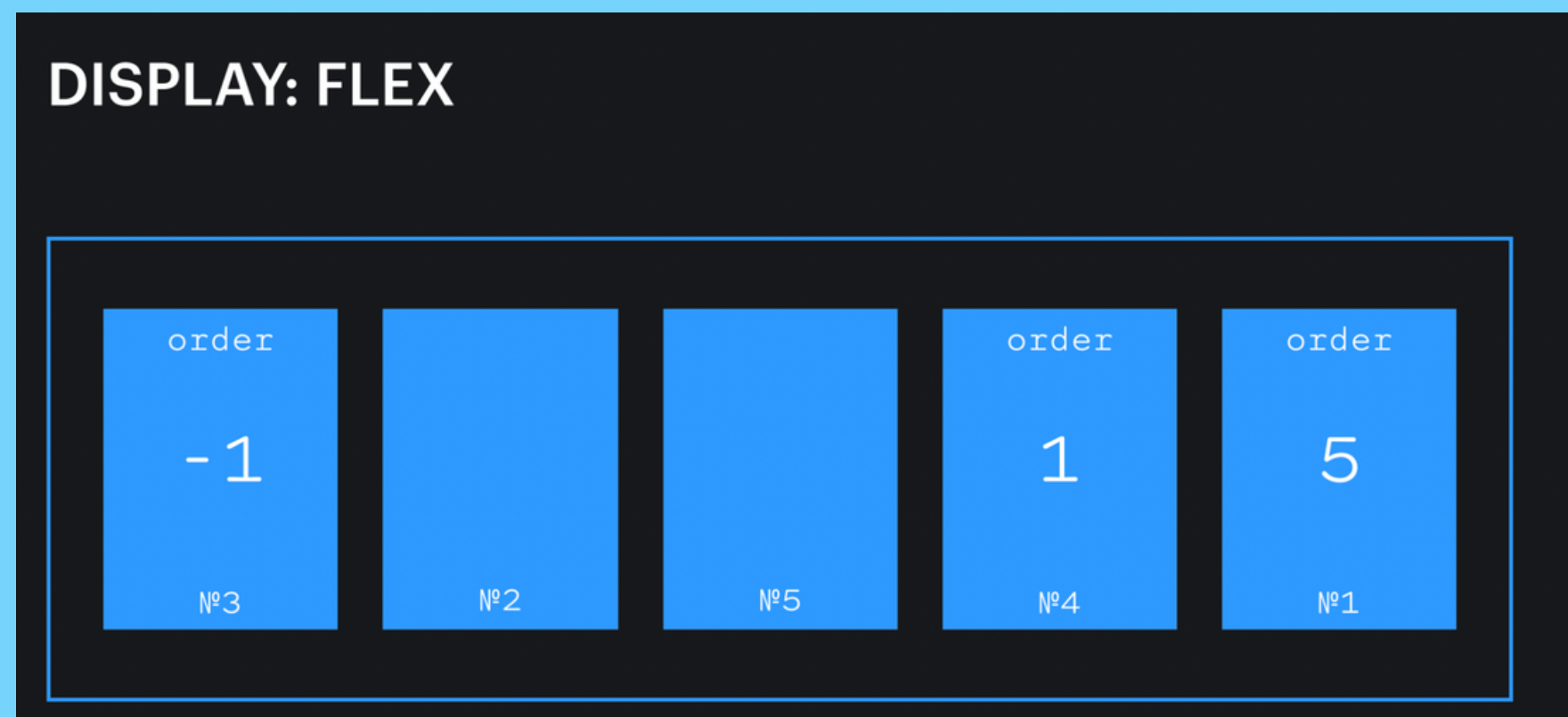
По умолчанию элементы отображаются в том порядке, в котором они расположены в разметке, а значение свойства `order` равно 0.

Значение задаётся в виде целого отрицательного или положительного числа. Элементы встают по возрастающей.

При помощи свойства `order` можно менять порядок отображения флекс-элементов внутри флекс-контейнера.

По умолчанию элементы отображаются в том порядке, в котором они расположены в разметке, а значение свойства `order` равно 0.

Значение задаётся в виде целого отрицательного или положительного числа. Элементы встают по возрастающей.



## flex-grow

Это свойство указывает, может ли вырастать флекс-элемент при наличии свободного места, и насколько. По умолчанию значение равно 0. Значением может быть любое положительное целое число (включая 0).

Если у всех флекс-элементов будет прописано flex-grow: 1, то свободное пространство в контейнере будет равномерно распределено между всеми.

Если при этом одному из элементов мы зададим flex-grow: 2, то он постарается занять в два раза больше свободного места, чем его соседи.

## flex-shrink

Свойство flex-shrink полностью противоположно свойству flex-grow. Если в контейнере не хватает места для расположения всех элементов без изменения размеров, то свойство flex-shrink указывает, в каких пропорциях элемент будет уменьшаться. Чем больше значение у этого свойства, тем быстрее элемент будет сжиматься по сравнению с соседями, имеющими меньшее значение. Значение по умолчанию — 1. Значением может быть любое целое положительное число (включая 0).

Два предыдущих свойства работают с пропорциональным разделением пространства, не с конкретными размерами. Они довольно непростые, даже опытный разработчик не всегда знает, как они в точности работают. Загляните в конец статьи, если хотите подробнее почитать о каждом из них.

## flex-basis

Свойство flex-basis указывает на размер элемента до того, как свободное место будет распределено (см. flex-grow).

Значением может быть размер в любых относительных или абсолютных единицах: 20rem, 5vw, 250px. А также можно использовать ключевое слово auto (значение по умолчанию). В этом случае при расчёте размера элемента будут приниматься во внимание значения свойств width, max-width, min-width или аналогичные свойства высоты, в зависимости от того, в каком направлении идёт основная ось.

Если никакие размеры не заданы, а свойству flex-basis установлено значение auto, то элемент занимает столько пространства, сколько нужно для отображения контента.

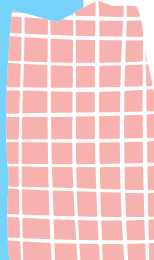
`flex`

Свойство-шорткат, с помощью которого можно указать значение трёх свойств одновременно: `flex-grow`, `flex-shrink` и `flex-basis`. Первое значение является обязательным, остальные опциональны. Значение по умолчанию: `0 1 auto`, что расшифровывается как `flex-grow: 0`, `flex-shrink: 1`, `flex-basis: auto`.

`align-self`

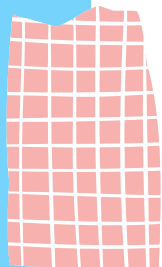
При помощи этого свойства можно выровнять один или несколько элементов иначе, чем задано у родительского элемента. Например, в коде выше у родителя задано выравнивание вложенных элементов по верхнему краю родителя. А для элемента с классом `.item` мы задаём выравнивание по нижнему краю.

# MEDIA QUERIES



Media queries в CSS позволяют применять стили на основе характеристик устройства или размера экрана, на котором отображается веб-страница. Они позволяют создавать адаптивные макеты, которые могут изменяться в зависимости от контекста просмотра.

Media queries используются для определения определенных точек разрыва или диапазонов ширины экрана, на которых применяются различные стили. Когда условие внутри media query выполняется, стили внутри блока медиа-запроса применяются к соответствующему элементу или элементам.

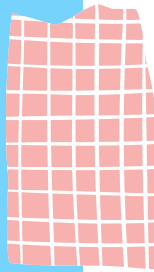


Они могут быть использованы для управления шрифтами, размерами, расположением элементов, отображением или скрыванием определенных частей содержимого и многим другим.

Применение media queries позволяет создавать адаптивные и отзывчивые веб-страницы, которые могут оптимально отображаться на различных устройствах и экранах.

Вы можете иметь один набор правил для компьютерных экранов, для принтеров, для портативных приборов, для типа телевизионных приборов, и так далее.





Медиа запросы могут использоваться для проверки многих вещей, таких как:

- ширина и высота окна просмотра
- ширина и высота устройства
- ориентация (планшет/телефон находится в альбомном или портретном режиме?)
- разрешение

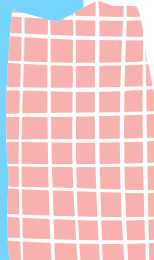
```
@media screen and (max-width: 768px) {  
    /* CSS стили для экранов с максимальной шириной 768px и меньше */  
    .text-info {  
        font-size: 14px;  
        color: red;  
    }  
}
```

```
- @media screen and (min-width: 768px) {  
  /* CSS стили для экранов с минимальной шириной 768px и больше */  
  .text-info {  
    width: 200px;  
    margin: 0 auto;  
  }  
}
```

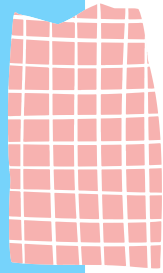
```
- @media screen and (min-width: 768px) and (max-width: 979px) {  
  /*@media screen and (min-width: 768px) and (max-width: 979px) указывает,  
  что стили, указанные внутри блока медиа-запроса, будут применяться только на устройствах  
  с шириной экрана между 768px и 979px, включительно.*/  
  .text-info {  
    font-weight: bold;  
    color: #89AF59;  
  }  
}
```

```
- @media screen and (orientation: portrait) {  
  /* CSS стили для портретной (вертикальной) ориентации экрана */  
  - .text-info {  
    |   display: none;  
  }  
  - }  
  - }
```

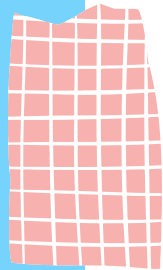
# TRANSFORM



Свойство transform в CSS позволяет изменять визуальное представление элементов путем применения различных преобразований. Оно предоставляет широкий спектр возможностей для трансформации элементов, включая перемещение, масштабирование, вращение, наклон и другие эффекты.



Преобразования, задаваемые с помощью свойства `transform`, применяются без изменения фактических размеров и расположения элемента в потоке документа. Они влияют только на отображение элемента, сохраняя его семантику и взаимодействие.



1. Перемещение (Translate): `translateX()`, `translateY()`, `translateZ()` позволяют перемещать элементы по горизонтали, вертикали и вдоль оси `Z` соответственно. Можно указывать значения в пикселях (`px`), процентах (`%`) или других единицах длины.
2. Масштабирование (Scale): `scaleX()`, `scaleY()`, `scaleZ()` позволяют масштабировать элементы по горизонтали, вертикали и вдоль оси `Z` соответственно. Значение `1` означает оригинальный размер, значения меньше `1` уменьшают размер, а значения больше `1` увеличивают размер элемента.
3. Вращение (Rotate): `rotate()` позволяет вращать элементы на указанный угол в градусах (`deg`) вокруг их центра. Положительные значения вращают по часовой стрелке, а отрицательные - против часовой стрелки.



**THANK  
YOU!**

Have a  
great day  
ahead.