

---

# МУЛЬТИМЕДИА

## Изображения

Изображения представляют собой замещаемый контент на странице, а это значит, что CSS не может влиять на их внутреннее устройство — только на позицию среди других элементов.

Тем не менее, существует достаточно возможностей управлять изображениями на странице. Начнем с общего.

### **border, opacity, box-shadow**

Изображениям, также как и блокам или тексту можно задавать настройки границ: скругление, толщина и тип границы.

```
.border{  
  padding: 5px;  
  border: 2px dotted #1f0365;  
  border-radius: 50% 20% / 10% 40%;  
}
```



```
.border{  
  padding: 5px;  
  border: 5px solid #c0c0c0;  
  border-top-right-radius: 20%;  
  border-bottom-right-radius: 20%;  
}
```

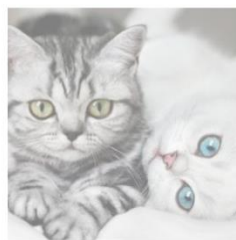
Управляем их прозрачностью с помощью свойства `opacity`. Значение от 0 до 1, где 0 – полностью прозрачный элемент, а 1 – непрозрачный.



**1**



**0.8**

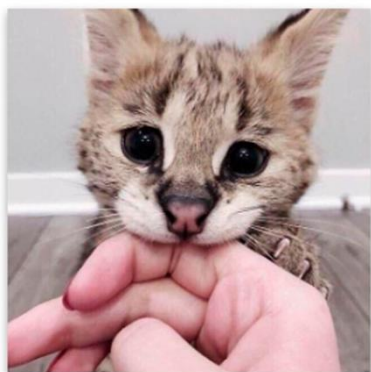


**0.5**



**0.2**

Задаем тень изображению как блоку с помощью `box-shadow`.



`box-shadow:`

```
rgba(0, 0, 0, 0.16) 0px 3px 6px,  
rgba(0, 0, 0, 0.23) 0px 3px 6px;
```

## Настройка размеров

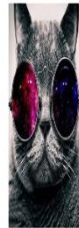
Задавая размеры через стили, вы должны внимательно следить за сохранением пропорций изображения.

Помните, что указанные одновременно значения **width** и **height** с большой вероятностью исказят ваше изображение.

А если вы установите только одно, второе подстроится автоматически.

```
img{
  width: 100px;
  height: 300px;
}
img{
  width: 150px;
  height: 200px;
}
img{
  width: 300px;
  height: 100px;
}
```

600\*600



100\*300



150\*200



300\*100

\*при создании контента ни один котик не пострадал

## object-fit

Свойство **object-fit** определяет, как содержимое заменяемого элемента, такого как **img** или **video**, должно заполнять контейнер относительно его высоты и ширины.

Если вы внимательно изучали предыдущие лекции, то заметите сходство возможных значений свойства с **background-size**.

**fill**. Смещаемый контент меняет свой размер таким образом, чтобы заполнить всю область внутри контейнера: используется вся ширина и высота блока.

600\*600



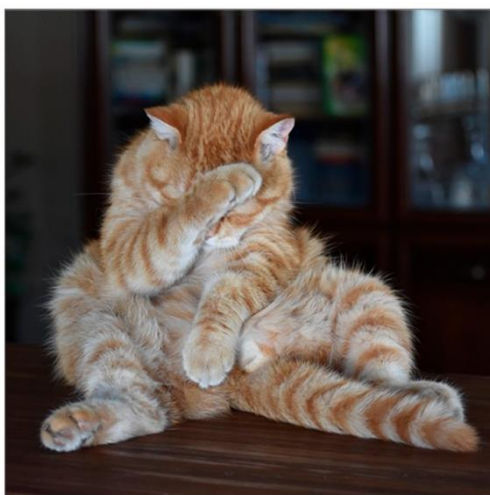
150\*200



*Если пропорции контейнера и изображения не совпадают, изображение будет искажено, но вписано в контейнер и по ширине и по высоте.*

**contain.** Смещаемый контент меняет свой размер таким образом, чтобы подстроиться под область внутри блока пропорционально собственным параметрам: окончательный размер контента будет определён как "помещённый внутрь" блока, ограничиваясь его шириной и высотой.

600\*600



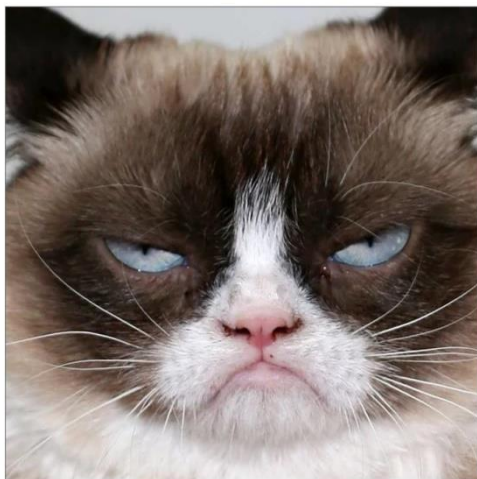
150\*200



*Если пропорции контейнера и изображения не совпадают, изображение будет показано целиком, но мы увидим фоновый цвет контейнера.*

**cover.** Смещаемый контент меняет свой размер таким образом, чтобы сохранять свои пропорции при заполнении блока: окончательный размер контента будет определён как "покрытие" блока, ограничиваясь его шириной и высотой.

600\*600



150\*200



*Если пропорции контейнера и изображения не совпадают, рамка изображения будет заполнена целиком, пропорции не искажутся. Изображение полностью впишется в контейнер по одной из сторон, но часть изображения будет обрезана.*

**none.** Смещаемый контент не изменяет свой размер с целью заполнить пространство блока: конечный размер контента будет определён с использованием алгоритма изменения размера по умолчанию, а также размер объекта по умолчанию равен ширине и высоте смещаемого контента.



600\*600

150\*200



*Если пропорции и размеры контейнера и изображения не совпадают, рамка изображения будет заполнена целиком, пропорции не искажутся, но вы увидите только часть изображения.*

**scale-down.** Контент изменяет размер, сравнивая разницу между none и contain, для того, чтобы найти наименьший возможный размер объекта.

**Важно!** Внешний вид изображения будет зависеть от исходных размеров и пропорций, а также от степени их несоответствия итоговому контейнеру.

### **object-position**

CSS-свойство **object-position** определяет выравнивание контента внутри блока элемента. На областях блока, не покрытых объектом замещаемого элемента, будет отображаться фон элемента.

Вы можете задавать способ подстройки собственного размера объекта замещаемого элемента (т. е., его естественного размера) под размер блока элемента, используя **object-fit**.

Используем от одного до четырёх значений (top, right, bottom, left), которые определяют 2D-позицию элемента. Могут

использоваться как абсолютные, так и относительные значения сдвигов.

Позиция может быть даже такой, что замещаемый элемент будет отрисовываться за пределами своего блока.

## object-fit + object-position



### image-orientation

Свойство CSS задает независимую от макета коррекцию ориентации изображения.

Предназначено только для корректировки ориентации изображений, снятых с повернутой камерой. Его не следует использовать для произвольных поворотов. Для любых целей, кроме исправления ориентации изображения в зависимости от того, как оно было снято или отсканировано, лучше использовать настройки трансформации,

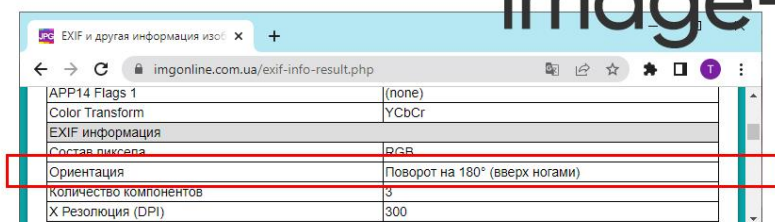
При использовании в сочетании с другими свойствами CSS, любое вращение применяется до всех остальных преобразований.

Свойство может принимать два значения.

**none** Не применяет никакого дополнительного поворота изображения; изображение ориентировано в соответствии с кодировкой или в соответствии с другими значениями свойств CSS.

**from-image** Начальное значение по умолчанию. Информация EXIF (метаданные), содержащаяся в изображении, используется для соответствующего поворота изображения.

## image-orientation



APP14 Flags 1	(none)
Color Transform	YCbCr
EXIF информация	
Состав пикселя	BGR
Ориентация	Поворот на 180° (вверх ногами)
Количество компонентов	3
X Резолюция (DPI)	300

from-image



none



Важно! Файлы примера для этого свойства тестируйте на сервере.

## image-rendering

Свойство CSS задает алгоритм масштабирования изображения. Свойство применяется к самому элементу, к любым изображениям, установленным в других его свойствах, и к его потомкам.



Браузер будет масштабировать изображение, когда автор страницы указывает размеры, отличные от его естественного размера. Масштабирование также может происходить из-за взаимодействия с пользователем. Это свойство **не влияет** на немасштабированные изображения.

**auto** Алгоритм масштабирования зависит от браузера. Последние версии используют алгоритмы для обеспечения максимального качества.

**smooth** Изображение должно быть масштабировано с помощью алгоритма, который максимизирует внешний вид изображения. В частности, допустимы алгоритмы масштабирования, которые «сглаживают» цвета. В большей степени режим предназначен для фотографий.

**crisp-edges** Изображение масштабируется с помощью алгоритма ближайшего соседа: цвет пикселя принимается равным цвету ближайшего к нему пикселя в исходном изображении.

**pixelated** Используя алгоритм ближайшего соседа, изображение масштабируется до следующего целого числа, кратного его исходному размеру или равного ему, а затем уменьшается до целевого размера, как для smooth.

crisp-edges



pixelated



## filter

Свойство CSS применяет к элементу графические эффекты, такие как размытие или изменение цвета. Фильтры обычно используются для настройки рендеринга изображений, фона и границ.

Свойство **filter** указывается как 0, одна или несколько функций, перечисленных ниже. Если параметр какой-либо функции недействителен, функция возвращает значение **none**.

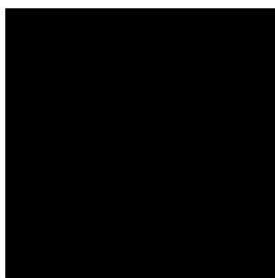
Если значения свойства содержат несколько функций, фильтры применяются по порядку.

**blur()** Применяет размытие по Гауссу к входному изображению.



**brightness()** Применяет линейный множитель к изображению, делая его более или менее ярким. Значение 0 создаст полностью черное изображение; 1 или 100% - оставит изображение без изменений. От 0 до 1 или менее 100% - уменьшат яркость. Больше 1 или более 100% добавляют яркость изображения.

`brightness(0);`



`brightness(1);`



`brightness(50%);`



`brightness(200%);`

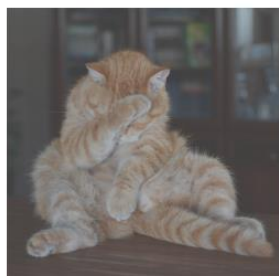


**contrast()** Регулирует контрастность входного изображения. Значение 0% делает изображение серым, 100% соответствует контрастности по умолчанию, а значения выше 100% повышают контраст.

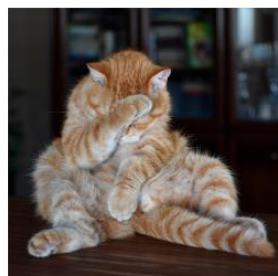
`contrast(0%);`



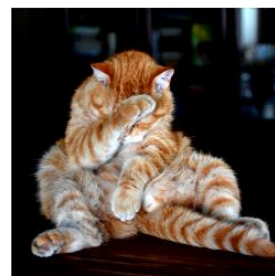
`contrast(50%);`



`contrast(100%);`

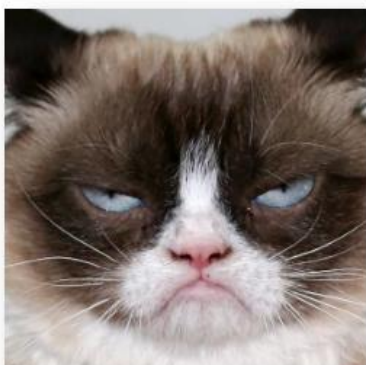


`contrast(150%);`

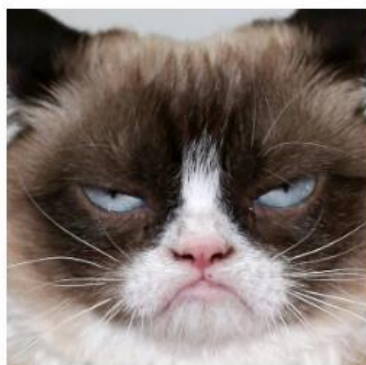


**drop-shadow()** Применяет параметр как тень, повторяющую контуры изображения. Любые фильтры после `drop-shadow()` применяются к тени.

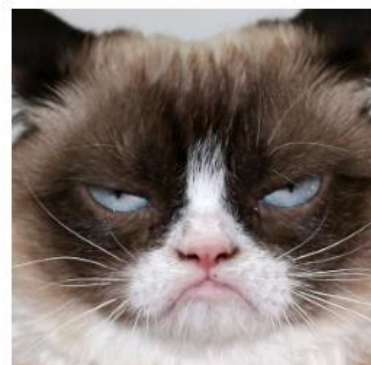
`drop-shadow  
(16px 16px 10px black);`



`drop-shadow  
(10px 10px 5px rgb(66, 9, 136));`



`drop-shadow  
(5px 5px 1px rgb(109, 109, 109));`



**grayscale()** Преобразует изображение в оттенки серого. Значение 100% преобразует изображение полностью в



градации серого. Начальное значение 0% оставляет изображение без изменений.

`grayscale(50%);`



`grayscale(75%);`



`grayscale(100%);`

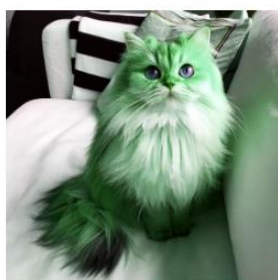


**hue-rotate()** Применяет поворот оттенка. Значение определяет количество градусов на цветовом круге, на которое будут скорректированы входные настройки. Значение 0deg оставляет вход без изменений.

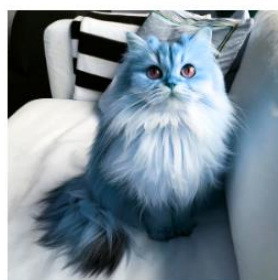
`hue-rotate(30deg);`



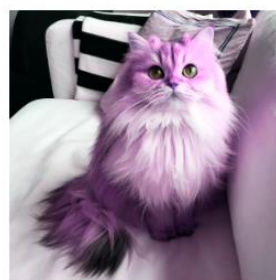
`hue-rotate(90deg);`



`hue-rotate(180deg);`



`hue-rotate(270deg);`



**invert()** Инвертирует настройки во входном изображении. Значение 100% полностью инвертирует изображение. Значение 0% оставляет его без изменений.

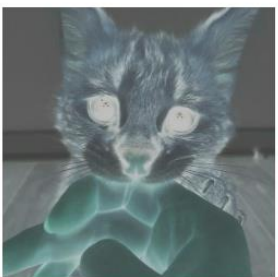
`invert(25%);`



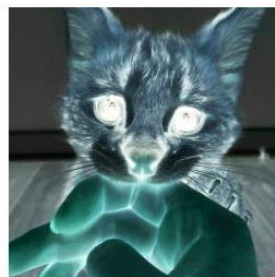
`invert(50%);`



`invert(75%);`



`filter: invert(140%);`



**opacity()** Применяет к изображению прозрачность. 0% делает изображение полностью прозрачным и 100% оставляет его без изменений.

opacity(25%);



opacity(50%);



opacity(75%);



opacity(100%);



**saturate()** Регулирует насыщенность изображения. 0% полностью ненасыщенное, 100% оставляет изображение без изменений, и более 100% увеличивает значения насыщенности.

saturate(0%);



saturate(50%);



saturate(100%);

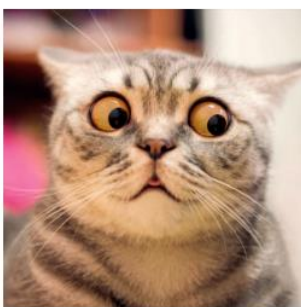


saturate(200%);

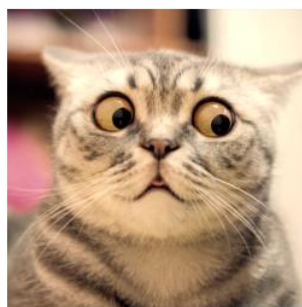


**sepia()** Добавляет эффект сепии к изображению. 100% для максимального эффекта и 0 чтобы оставить изображение без изменений.

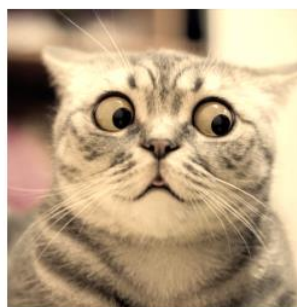
sepia(25%);



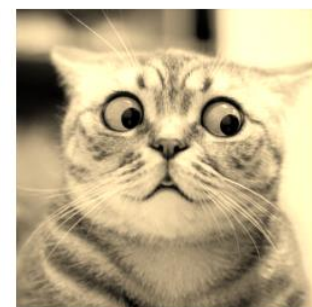
sepia(50%);



sepia(75%);



sepia(100%);



Объединение фильтров



Вы можете комбинировать любое количество фильтров, перечисляя их в качестве значения свойства. Фильтры применяются в заявленном порядке.

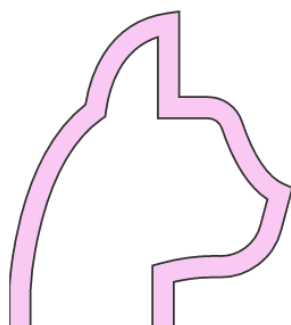
## svg

Настройка и стилизация svg элемента может заключаться в редактировании параметров, настраиваемых внутри него самого.

Атрибуты, определяющие внешний вид создаваемых фигур

- stroke – граница
- stroke-width – толщина границы
- fill - заливка

Если вам нужно изменить какое-то из значений с помощью стилей, вынесите их из элемента. Будьте внимательны, ваше изображение состоит из svg обертки и path контура.



---

## IMAGE PLACEHOLDER

Мы уже знаем, за что отвечает атрибут **placeholder** текстового поля. Он выводит предопределенный текст внутри поля формы, который исчезает при получении фокуса или при наборе текста. Обычно отображается серым цветом.

```
<input type="text" placeholder="текст">
```

Что такое псевдоэлемент **::placeholder** разбирали на прошлом занятии. Стилль placeholder-а можно изменить с помощью CSS правил:

```
input[type="search"]::placeholder {color: # ffd595;}
```

При этом очень часто возникает необходимость расположить контент на странице еще до того как он окончательно утвержден. С текстом нам поможет «рыба». Вспоминаем функцию `lorem()`.

Но что же делать с изображениями? Кому хочется тратить время на поиск временных картинок нужного размера или подготовку их в графическом редакторе? Нам помогут сервисы, предоставляющие подобные изображения в готовом виде. Представляем некоторые из них.

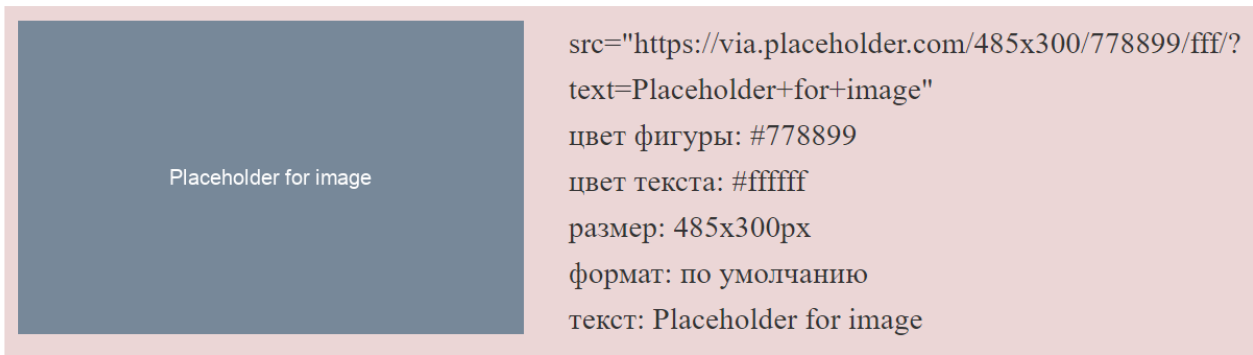
**[https://via.placeholder.com/ \(sample-1.html\)](https://via.placeholder.com/(sample-1.html))**

При попытке перехода по этой ссылке вы увидите ошибку. Но нам нужен не сам сайт, а генерируемые им изображения.

Настройки изображения задаются через / как адрес изображения в следующем порядке.

- **Размер** – одним (ширина и высота одинаковые) или двумя числами (ширина и высота) в пикселях.

- **Цвет фигуры** – в формате трех или шести цифр (код без #).
- **Цвет текста** – аналогично цвету фигуры.
- **Текст** – с приставкой **?text=[ваш+текст]** плюсами заменяем пробелы.



[http://placekitten.com/ \(sample-2.html\)](http://placekitten.com/sample-2.html)

Быстрый и простой сервис для получения изображений котят для использования в качестве заполнителей в ваших проектах или коде. Просто укажите размер изображения (ширину и высоту) после URL-адреса, и вы получите фото-заполнитель.

<http://placekitten.com/200/300> или

<http://placekitten.com/g/200/300>

Указатель **g** подберет для вас изображение в черно-белом варианте.

Аналогично работает сервис для получения фото медведей

<https://placebear.com> или бородатых мужчин

<https://placebeard.it/>

! т с е те ст т  
 . те т , т к к -т с е

,

.

---

## ЗАДАНИЕ 19 ГАЛЕРЕЯ

Галереи изображений – страницы с большим количеством фото. Это могут быть полноценные изображения или их миниатюры. Такие мини-фото могут быть кликабельны, чтобы открыть полную версию изображения. Расположение фотографий зависит от ваших умений и фантазии.

Еще один вариант представления изображений в галерее – слайдеры. **Слайдер** – это специальный элемент веб-дизайна, представляющий собой блок определенной ширины. Главная его фишка в изменяющихся в ручном или автоматическом режиме элементах – картинка, текст или ссылки. Попробуем реализовать необычную галерею, используя псевдоклассы и свойства Flexbox-ов.

1. Сначала подготовим структуру страницы. Это будет блок `.container` с заголовком. Выровняем его по центру страницы и установим ширину 90%.
2. Можете задать для страницы фон и цвет шрифта, а для заголовка выравнивание и размер.
3. В обертку с классом `.gallery__flex` поместим наши изображения. Для этого воспользуемся сайтом [loremflickr.com](https://loremflickr.com).

```
<div class="gallery__item">
  
</div>
<div class="gallery__item">
  
</div>
<div class="gallery__item">
  
</div>
<div class="gallery__item">
  
```

```

</div>
<div class="gallery__item">
  
</div>
<div class="gallery__item">
  
</div>

```

4. Стили будут состоять из нескольких частей. Устанавливаем статичную высоту блока и гибкое содержимое.

```

.gallery__flex {
  height: 600px;
  display: flex;
}

```

5. Изначально для всех дочерних блоков с изображениями задаем flex: 1; , то есть все блоки занимают одинаковое количество доступного пространства в родительском блоке. При наведении на определённый блок с изображением, меняем ему значение flex, например на flex: 16;. Теперь элемент при наведении будет занимать в 16 раз больше пространства, чем остальные элементы

```

.gallery__item { flex: 1; }
.gallery__item:hover { flex: 16; }

```

6. Пропорционально растягиваем изображение на весь блок gallery\_\_item.

```

.gallery__item img {
  width: 100%;
  height: 100%;
  object-fit: cover;
  display: block;
}

```

7. Попробуйте самостоятельно дописать код, чтобы наша «гармошка» из вертикальной стала горизонтальной.