**Документация библиотеки для кадрирования, изменения размера и положения HTML изображения на языке JavaScript**

**Главный разработчик:**

Федотов М.А.

2023

СОДЕРЖАНИЕ

[**1.** **Предварительное описание** 3](#_Toc137036623)

[**2.** **Структура проекта** 3](#_Toc137036624)

[**3.** **Рамки системы** 3](#_Toc137036625)

[**4.** **Жизненный цикл экземпляра класса** 5](#_Toc137036626)

[**5.** **Пользовательский интерфейс класса библиотеки (описание операций)** 5](#_Toc137036627)

[**6.** **Интерфейс класса** 6](#_Toc137036628)

[**7.** **Словарь определений** 14](#_Toc137036629)

1. **Предварительное описание**

Плагин для кропа и обрезки изображений. Плагин должен по функционалу работать также как работает кроп на сервисах fotor.com или canva.com (сайт доступен только с vpn из РФ). Плагин должен корректно работать с повернутыми изображениями на произвольный угол. Должен быть доступен одновременно и ресайз самого изображения и изменение размера области кропа, а также перемещение этой области.Допускается как самописный плагин, так и доработка существующих, например cropperjs. Не допускается использование фреймворков (vuejs, angular и т.п.). Плагин должен быть написан нативно или с использованием jquery. Должен быть читаемый код с возможностью дополнения или модификации кода. Функционал обрезки/ресайза через canvas или модификация исходного изображения приобрезке не требуется. Плагин должен возвращать координаты и размер нового изображения, а дальше исходное изображение смещаться через transform: translate или само изображение через object-position свойство css или просто через left/top. Само исходное изображение никак не должно модифицироваться при обрезке. Всегда должна была возможность изменить обрезку/размер изображения при обрезке (у fotor.com и canva.com работает именно по такому принципу). Прикладываю файл crop.html. В нем размещено изображение для примера. После написания плагина нужно продемонстрировать его работу на примере этого файла. Также необходимо чтобы плагин не затрагивал структуру DOM внутри #container, а клонировал исходное изображение и скрывал его на время работы, а сам плагин добавлялся в body. По завершению обрезки нужно чтобы плагин удалялся из body, модифицировал стили #image и/или само изображение. Не имеет значения каким образом будет выполнена кнопка сохранения изменений/отмены. Достаточно чтобы плагин запускался по двойному клику по изображению. А сохранялись изменения по клику на любую кнопку/через горячую клавишу. Нужно чтобы в будущем можно было это поменять под свои нужды.

1. **Структура проекта**

* CMSLibrary.js – файл библиотеки

1. **Рамки системы**

* Библиотека поддерживает спецификацию JavaScript ECMAScript 6
* Один экземпляр класса (изображение) содержит в себе три HTML элемента вложенных друг в друга:
  + База изображения (основа);
  + Активный блок (элемент, который обрезает изображение путём CSS);
  + Изображение;
* HTML элементы экземпляра класса не могут иметь никаких других свойств трансформации, или тегов изменяющих их форму.

Исключения:

* + Активный блок

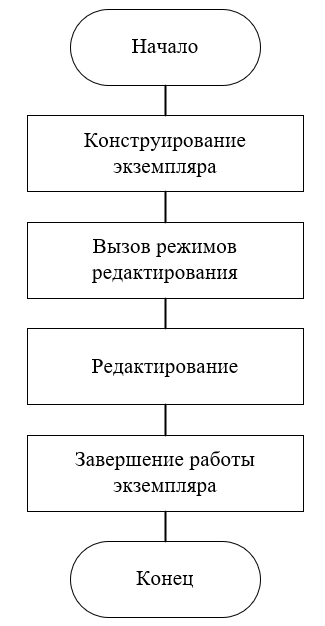
inline-CSS свойства width, height и transform (только со свойством rotate);

* + Изображение

inline-CSS свойства width, height;

* Библиотека не генерирует элементы экземпляра класса, они уже должны быть готовы перед вызовом конструктора;
* База изображения должна иметь DIV html-тэг
* Активный блок должен иметь DIV html-тэг
* Изображение должно иметь IMG html-тэг
* Библиотека использует собственные классы стилей, имена которых начинаются с “cmr-”;
* Библиотека не предоставляет визуального интерфейса для включения того или иного режима или события внутри библиотеки
* Библиотека хранится в глобальной области под именем “CMRImage”
* Скрипт библиотеки должен быть запущен до взаимодействия с ней
* Библиотека работает на web-страницах
* Информация о положении элементов изображения храниться в inline-CSS стилях
* Ссылка на экземпляр класса сохраняется в базе изображения в качестве свойства “\_cmr”
* Активный блок должен иметь CSS свойство “overflow: hidden;”
* Элемент изображение должно иметь CSS свойство “position: relative;”

1. **Жизненный цикл экземпляра класса**



Блок схема №1 – жизненный цикл экземпляра класса CMRImage

1. **Пользовательский интерфейс класса библиотеки (описание операций)**

**CMRImage** – основной класс библиотеки создающий экземпляр;

1. **set –** заменяет элемент базы изображения;
2. **get –** возвращает HTML элемент базы изображения;
3. **setMode** – устанавливает режим редактирования;
4. **clear** – отчищает все следствия установки режимов (отчищает все слушатели событий и сгенерированные библиотекой элементов конкретного экземпляра);
5. **dispose –** завершает работу с изображением. Удаляется ссылка на экземпляр из базы изображения. Отчищает все слушатели событий и сгенерированные библиотекой элементов конкретного экземпляра с сохранением результата;
6. **getGeometry** – получение текущего состояние изображения;
7. **isDisposed** – проверят экземпляр на завершённость;
8. **Интерфейс класса**

**CMRImage** – основной класс библиотеки, отвечающий за инициализацию экземпляра (конструктор).

**Аргументы:**

HTMLDivElement – база изображения

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

* **set(HTMLDivElement)**

Метод заменяет элемент базы изображения

**Аргументы:**

HTMLDivElement – база изображения

**Возвращает:**

undefined

**Пример:**

const instance = new CMRImage();

const base = document.querySelector(“#image-base-1”);

instance.set(base);

* **get()**

Метод возвращает HTML элемент базы изображения.

**Возвращает:**

HTMLDivElement

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

console.log(instance.get());

* **setMode(String)**

Метод устанавливает режим редактирования

**Аргументы:**

String

Возможные режимы:

* “crop”
* “resize”

**Возвращает:**

undefined

**Пример:**

const instance = new CMRImage();

const base = document.querySelector(“#image-base-1”);

instance.set(base);

instance.setMode(“crop”);

* **clear()**

Метод отчищает все следствия установки режимов (отчищает все слушатели событий и сгенерированные библиотекой элементов конкретного экземпляра) с сохранением результата.

**Возвращает:**

undefined

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

instance.setMode(“crop”);

instance.clear();

* **dispose()**

Метод, завершающий работу с изображением. Удаляется ссылка на экземпляр из базы изображения. Отчищает все слушатели событий и сгенерированные библиотекой элементов конкретного экземпляра с сохранением результата.

**Возвращает:**

undefined

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

instance.dispose();

* **getGeometry()**

Метод, возвращающий текущие параметры изображения для дальнейшей обработки

**Возвращает:**

Object {

activeWidth: Number,

activeHeight: Number,

activeX: Number,

activeY: Number,

imageWidth: Number,

imageHeight: Number,

imageX: Number,

imageY: Number,

angle: Number

}

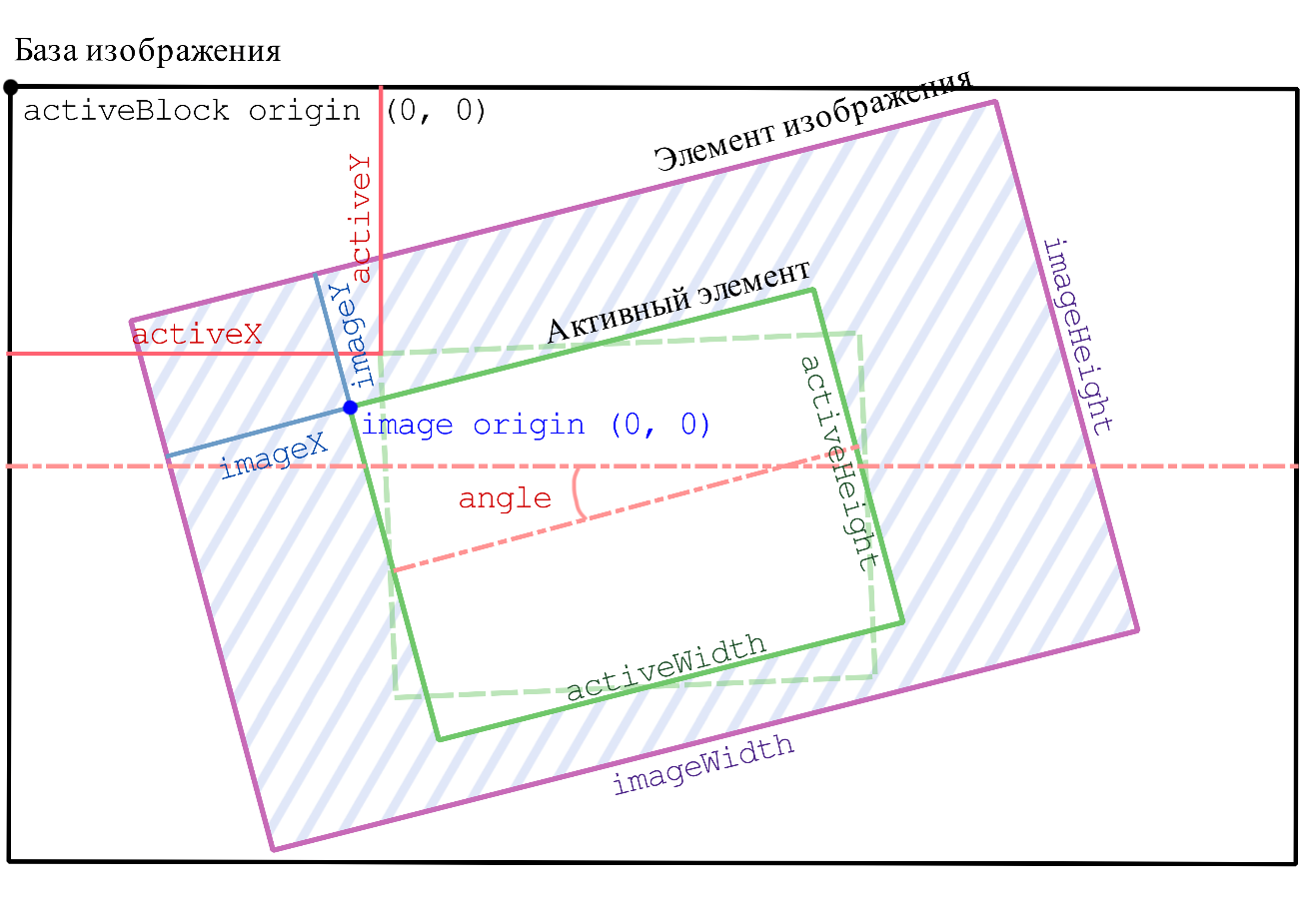


Рисунок №1 – Значение свойств размещения

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

console.log(instance.getGeometry());

* **isDisposed()**

Метод, проверяющий экземпляр на завершённость.

**Возвращает:**

Bool

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

console.log(instance.isDisposed());

* **\_createResize()**

Приватный метод, устанавливающий слушатели событий и генерирующий HTML элементы управления размером изображения

**Возвращает:**

undefined

* **\_createCrop()**

Приватный метод, устанавливающий слушатели событий и генерирующий элементы управления кадрированием изображения

**Возвращает:**

undefined

* **\_gen(String tag[, Object options[, Bool/HTMLElement parent[, Bool/Function init]]])**

Приватный метод, генерирующий элемент HTML

**Аргументы:**

* tag (String) – HTML тэг создаваемого элемента
* options (Object, опциональный)– свойства создаваемого элемента (в том числе дети)
* parent (Bool/HTMLElement, опциональный) – родитель в которого будет добавлен элемент
* init (Bool/Function(HTMLElement), опциональный) – функция, обрабатывающая элемент перед добавлением его в родителя.

**Возвращает:**

HTMLElement

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

console.log(instance.\_gen(“div”, {

“style”: “color: white”,  
 “contenteditable”: true,

“textContent”: “test”  
}, document.body, function(element) {

Element.textContent = “complete”

}));

* **\_setGeometry(Object)**

Приватный метод, устанавливающий состояние изображения записанного в атрибуте **state.geometry**

**Аргументы:**

Object {

?activeWidth: Number,

?activeHeight: Number,

?activeX: Number,

?activeY: Number,

?imageWidth: Number,

?imageHeight: Number,

?imageX: Number,

?imageY: Number,

?angle: Number

}

**Возвращает:**

undefined

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

instance.\_setGeometry({

imageX: 50,

imageY: 25,

});

* **\_update()**

Приватный метод, обновляющей состояние стилей элементов экземпляра

**Возвращает:**

Undefined

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

instance.\_setGeometry ({

imageX: 50,

imageY: 25,

});

instance.\_update();

* **\_render(Object)**

Приватный метод, устанавливающий состояние изображения записанного в атрибуте **state.geometry** и обновляющий состояние стилей.

**Аргументы:**

Object {

?activeWidth: Number,

?activeHeight: Number,

?activeX: Number,

?activeY: Number,

?imageWidth: Number,

?imageHeight: Number,

?imageX: Number,

?imageY: Number,

?angle: Number

}

**Возвращает:**

undefined

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

console.log(instance.\_gen(“div”, {

“style”: “color: white”,  
 “contenteditable”: true,

“textContent”: “test”  
}, document.body, function(element) {

Element.textContent = “complete”

}));

* **\_setGeometry(Object)**

Приватный метод, устанавливающий состояние изображения записанного в атрибуте **state.geometry**

**Аргументы:**

Object {

?activeWidth: Number,

?activeHeight: Number,

?activeX: Number,

?activeY: Number,

?imageWidth: Number,

?imageHeight: Number,

?imageX: Number,

?imageY: Number,

?angle: Number

}

**Возвращает:**

undefined

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

instance.\_setGeometry({

imageX: 50,

imageY: 25,

});

* **\_update()**

Приватный метод, обновляющей состояние стилей элементов экземпляра

**Возвращает:**

Undefined

**Пример:**

const base = document.querySelector(“#image-base-1”);

const instance = new CMRImage(base);

instance.\_render({

imageX: 50,

imageY: 25,

});

* **\_parse()**

Приватный метод, записывающий текущее состояние элементов в атрибут **state.geometry**

**Возвращает:**

undefined

* **state**

Атрибут, содержащий в себе текущую состояние экземпляра

state {

Object geometry

HTMLDivElement base

…

}

1. **Словарь определений**

**Библиотека** – коллекция предварительно написанного кода JavaScript, который предоставляет повторно используемые функции, компоненты и утилиты для упрощения задач веб-разработки.

**Экземпляр** – объект, который создается на основе определенной схемы классов или функции конструктора. Когда класс определен в JavaScript, он служит шаблоном или основой для создания нескольких экземпляров этого класса.

**Класс** – это схема или шаблон, который описывает структуру и поведение объектов. Он служит основой для создания нескольких экземпляров класса, которые представляют собой отдельные объекты со своими собственными уникальными свойствами и методами.

**База изображения** – HTML элемент с тэгом DIV, являющийся статичной основой для любого экземпляра изображения, содержащий в себе единственный дочерний элемент – *активный блок*.

**Активный блок** – HTML элемент с тэгом DIV, являющийся рамкой кадрирования изображения и, также, использующийся для установления наклона изображения. Содержит в себе единственный элемент изображения.

**Элемент изображения (изображения)** – HTML элемент с тэгом IMG к которому применяются изменения размера и кадрирования

**Экземпляр изображения** – совокупность HTML элементов, хранящихся в экземпляре класса.

**Inline-стиль** – CSS стиль записанный как атрибут HTML элемента.

**HTML элемент** – элемент записанный в разметке HTML

**Тэг** – HTML тэг, отображающий вид блока разметки

**Класс стилей** – HTML атрибут, использующийся для идентификации повторяющихся HTML элементов для применения к ним CSS стилей

**CSS стиль** – это язык стилей, используемый для описания представления и визуального стиля HTML-элементов на веб-странице. Стили CSS определяют, как должны отображаться элементы, включая их расположение, цвета, шрифты, размеры и другие визуальные свойства.

**Метод** – это блок кода, состоящий из ряда инструкций, который можно вызывать по его имени. Он обязательно содержит возвращаемый тип, название, аргументы и тело метода.

**Атрибут** – это имена переменных вне функций и имена функций.