

Разработка веб-приложения для художественной обработки изображений с использованием HTML, CSS и JavaScript

Языков Кирилл Александрович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье представлена разработка интерактивного веб-приложения «Художник: Раскрась Мир» — инструмента для автоматической генерации художественных раскрасок на основе загруженного пользователем изображения. Приложение реализует 10 уникальных стилей преобразования изображений, включая карандашные наброски, контурные рисунки, пиксель-арт и имитацию старинной гравюры. Все функции реализованы с использованием нативных веб-технологий (HTML5, CSS3, JavaScript) и дополнены библиотеками FileSaver.js и JSZip для экспорта результатов. Пользователь может настраивать яркость, контраст, чёткость и прорисовку для каждого стиля отдельно, а также скачивать как отдельные изображения, так и весь комплект в виде ZIP-архива. Веб-приложение не требует подключения к интернету после загрузки и может использоваться как творческий инструмент, методическое пособие по цифровой графике или образовательный ресурс по клиентской веб-разработке.

Ключевые слова: HTML, CSS, JavaScript, обработка изображений, художественные фильтры, Canvas API, FileSaver.js, JSZip, веб-приложение.

Development of a Web Application for Artistic Image Processing Using HTML, CSS and JavaScript

Yazykov Kirill Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article presents the development of an interactive web application titled “Artist: Color the World” — a tool for automatically generating artistic coloring pages from a user-uploaded image. The application implements 10 unique image transformation styles, including pencil sketches, contour drawings, pixel art, and simulated vintage engravings. All functionality is built using native web technologies (HTML5, CSS3, JavaScript) and enhanced with the FileSaver.js and JSZip libraries for result export. Users can independently adjust brightness, contrast, sharpness, and detail rendering for each style and download either individual images or the entire set as a ZIP archive. The web application operates entirely offline after initial loading and can

serve as a creative tool, a methodological resource for digital graphics, or an educational aid in client-side web development.

Keywords: HTML, CSS, JavaScript, image processing, artistic filters, Canvas API, FileSaver.js, JSZip, web application.

1 Введение

1.1 Актуальность

Современные веб-технологии превратили браузер в мощную платформу для разработки интерактивных приложений, включая сложные графические инструменты, не требующие установки дополнительного ПО. Благодаря стандартам HTML5, CSS3 и JavaScript, а также таким API, как Canvas, стало возможным реализовывать на стороне клиента полноценные решения для обработки изображений, анимации и мультимедиа. Это особенно важно для образовательных, творческих и массовых приложений, где ключевыми требованиями являются доступность, кроссплатформенность и простота использования.

Особую значимость приобретают веб-сервисы, сочетающие художественную выразительность с технической автономностью — например, генераторы раскрасок, стилизаторы изображений или имитаторы графических техник. Такие приложения востребованы среди педагогов, детей, дизайнеров и широкой аудитории пользователей, поскольку позволяют мгновенно получать персонализированный визуальный контент прямо в браузере. При этом вся обработка данных происходит локально, что гарантирует конфиденциальность и безопасность. Разработка подобных решений на чистом JavaScript служит не только практическим инструментом, но и ценным учебным примером, демонстрирующим возможности клиентской веб-разработки в области цифровой графики и интерактивного дизайна.

1.2 Обзор исследований

С. А. Аксенова в своей работе рассмотрела общие принципы разработки веб-сайтов с использованием базовых клиентских технологий — HTML, CSS и JavaScript, уделив особое внимание структурированию контента, визуальному оформлению и реализации интерактивных элементов [1]. Ю.В.Белый и М. И. Жадан представили практический пример создания тематического веб-сайта, посвящённого лыжной акробатике, где продемонстрировали применение HTML5, CSS3 и JavaScript для построения адаптивного интерфейса и динамических компонентов в спортивной сфере [2]. Д. Дакетт в учебном пособии подробно и доступно изложил основы веб-дизайна, охватив как создание семантической структуры страниц с помощью HTML, так и их стилизацию средствами CSS, что делает издание ценным ресурсом для начинающих разработчиков [3]. Е. В. Козловская и Д.С.Кузьменков описали разработку образовательного веб-сайта для тестирования знаний по программированию, реализованного на современных веб-стандартах и обеспечивающего интерактивную проверку ответов и удобный пользовательский опыт [4]. И. Н. Ефремова, Е. А. Петрик и

А.М.Кириллов исследовали архитектурные подходы к созданию программно-информационной системы, направленной на автоматизацию и упрощение процесса конструирования веб-сайтов за счёт модульности и гибкой настройки компонентов [5].

1.3 Цель исследования

Целью исследования является разработка полнофункционального веб-приложения для художественной обработки изображений с возможностью настройки параметров с использованием HTML, CSS и JavaScript.

2 Материалы и методы

Для разработки приложения использовалась интегрированная среда Visual Studio Code, поддерживающая подсветку синтаксиса и отладку для веб-технологий. Архитектура приложения реализована в соответствии с принципом разделения ответственности:

- HTML5: определяет структуру страницы: загрузка изображения, область предпросмотра, сетка результатов, описание стилей.
- CSS3: отвечает за адаптивный дизайн, цветовую палитру, анимации и визуальное оформление элементов управления.
- JavaScript: реализует логику обработки изображений через Canvas API, включая применение фильтров, управление слайдерами и экспорт данных.

Дополнительно подключены две внешние библиотеки:

- FileSaver.js — позволяет сохранять сгенерированные изображения на устройство пользователя через API Blob.
- JSZip — формирует ZIP-архив из нескольких изображений для удобного пакетного скачивания.

3 Результаты

Веб-приложение «Художник: Раскрась Мир» создано для автоматического преобразования произвольного изображения в 10 различных художественных стилей, подходящих для раскрашивания (рис. 1). Оно предназначено для широкого круга пользователей: от детей и родителей до педагогов и дизайнеров. Приложение работает полностью автономно, не отправляет данные на сервер и не требует регистрации.

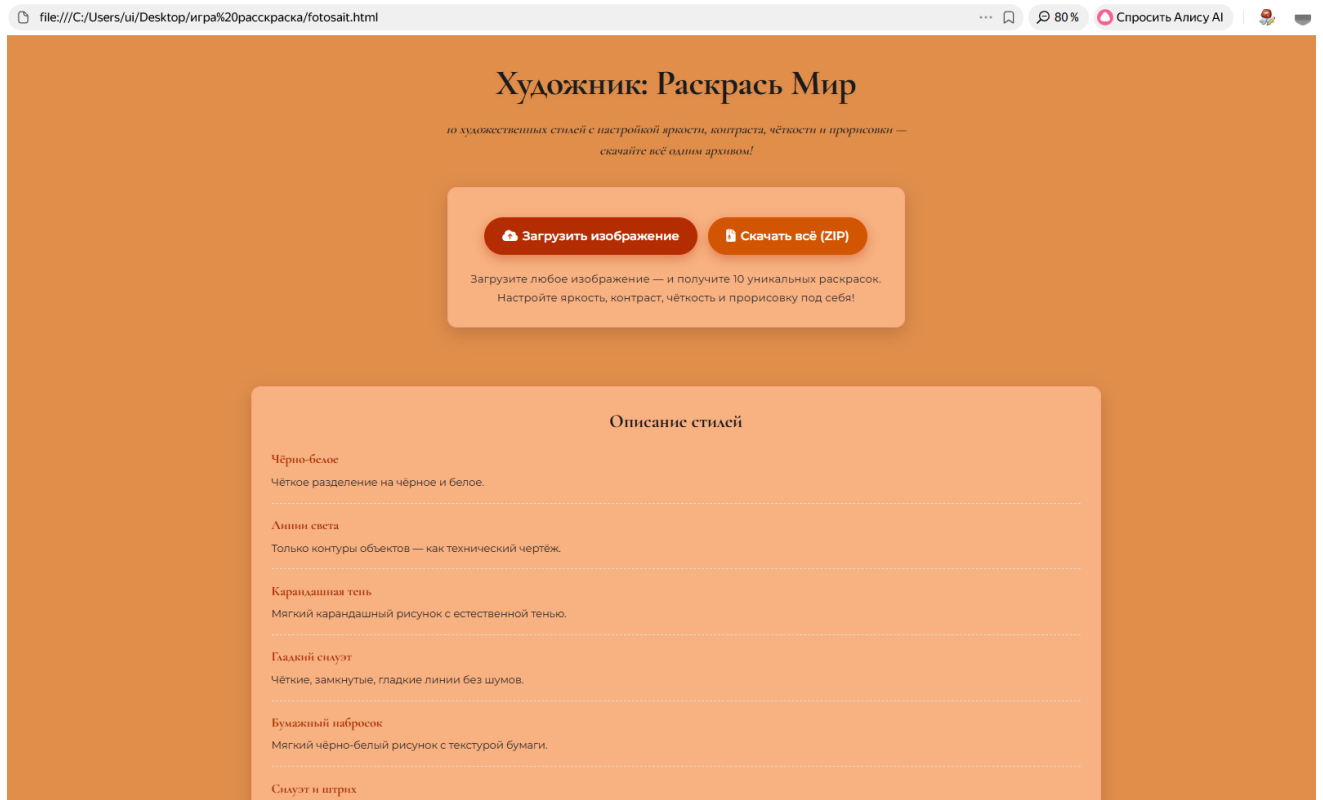


Рисунок 1 – Веб-приложение «Художник: Раскрась Мир»

Основные функции веб-сайта:

- 1) Загрузка изображения любого формата (JPEG, PNG и др.) (рис.2).
- 2) Автоматическая генерация 10 раскрасок в разных стилях.
- 3) Индивидуальная настройка яркости, контраста, чёткости и прорисовки для каждого стиля.
- 4) Предварительный просмотр всех результатов в сетке.
- 5) Скачивание одного изображения или всего набора в ZIP-архиве.

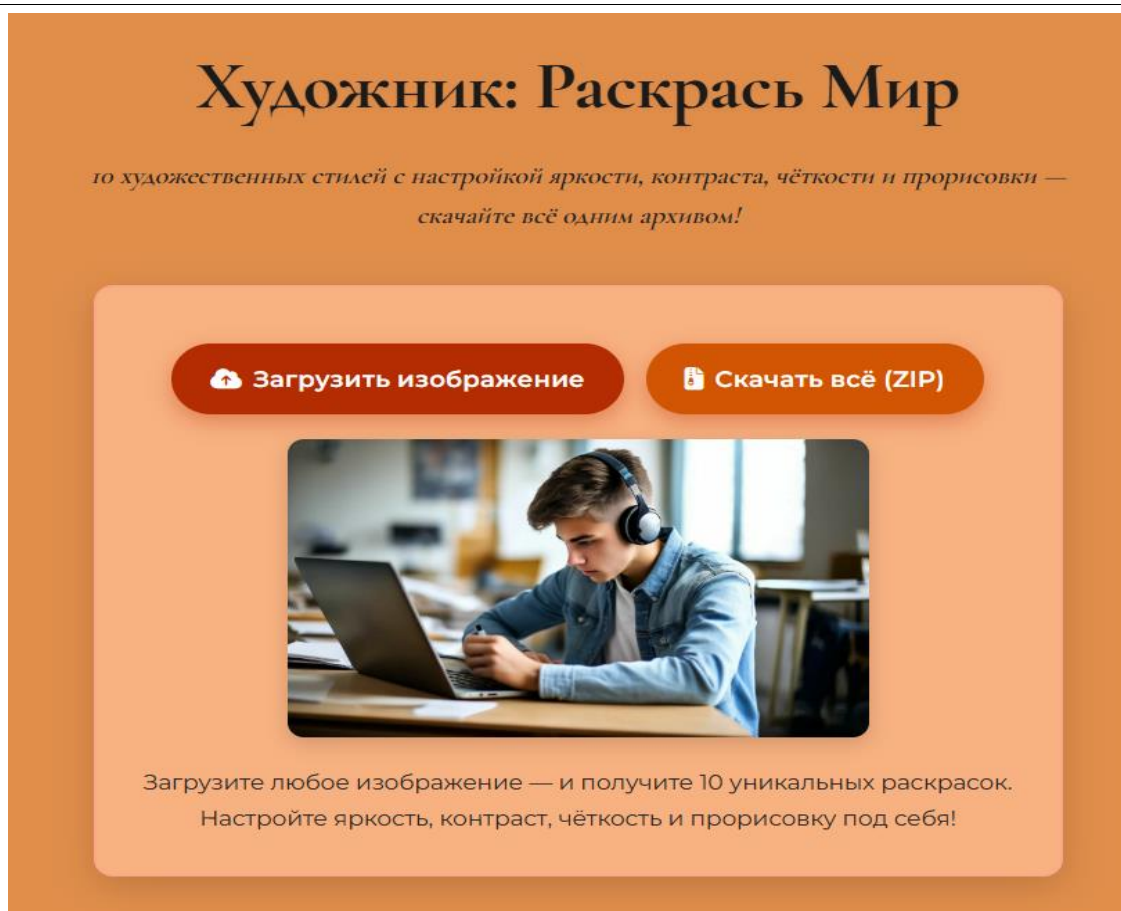


Рисунок 2 – Загрузка изображения

Для реализации функциональности экспорта изображений в разработанном веб-приложении используются две внешние JavaScript-библиотеки: FileSaver.js и JSZip. Обе подключаются через CDN и не требуют установки или сборки проекта, что обеспечивает простоту развёртывания и автономную работу приложения.

Библиотека FileSaver.js (версия 2.0.5) предоставляет кросс-браузерный API для сохранения файлов непосредственно на устройстве пользователя. В приложении она используется для реализации функции `saveAs()`, которая позволяет скачивать отдельные раскраски в формате PNG. Благодаря этой библиотеке обходятся ограничения стандартного поведения браузера, связанного с невозможностью напрямую сохранять Blob-данные без участия сервера.

Библиотека JSZip (версия 3.10.1) предназначена для создания, чтения и редактирования ZIP-архивов прямо в памяти браузера. В контексте данного проекта она применяется для объединения всех десяти сгенерированных изображений в один архив, который затем передаётся в FileSaver.js для скачивания. Это значительно упрощает пользователю процесс массового экспорта результатов.

Подключение библиотек осуществляется в HTML-документе следующим образом:

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/jszip/3.10.1/jszip.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/2.0.5/FileSaver.min.js"></s
cript>
```

Основным функционалом веб-приложения «Художник: Раскрась Мир» является обработка изображений, для чего было реализовано 10 различных художественных стилей (рис. 3). Каждый из десяти стилей решает определённую художественную или педагогическую задачу — от простого контурного рисунка до сложной имитации старинной печатной графики. Все стили реализованы программно с использованием Canvas API и не требуют внешних ресурсов. Для удобства анализа и воспроизводимости результатов ниже представлена таблица, в которой сопоставлены название стиля, его функциональное назначение и соответствующая JavaScript-функция, отвечающая за преобразование.

Таблица 1. Художественные стили

№	Название стиля	Описание	Функция в коде
1	Чёрно-белое	Бинаризация изображения по порогу 128	applyThreshold
2	Линии света	Обнаружение границ методом Собеля	applyEdgeDetect
3	Карандашная тень	Эффект карандашного рисунка через инверсию и размытие	applySketch
4	Гладкий силуэт	Чёткие, сглаженные контуры объектов	applyCleanOutline
5	Бумажный набросок	Серый рисунок с добавлением шума текстуры бумаги	applyPencilSketch
6	Силуэт и штрих	Комбинация карандашного рисунка и контурных линий	applyContourSketch
7	Форма и деталь	Совмещение силуэта и внутренних границ	applyContourEdges
8	Старинная печать	Имитация гравюры через штриховку переменной плотности	applyEngraving
9	Ретро-пиксели	Уменьшение разрешения и увеличение блоков до 8×8 пикселей	applyPixelArt
10	Элегантный контур	Тонкие, чистые линии без заливки	applyThinLine

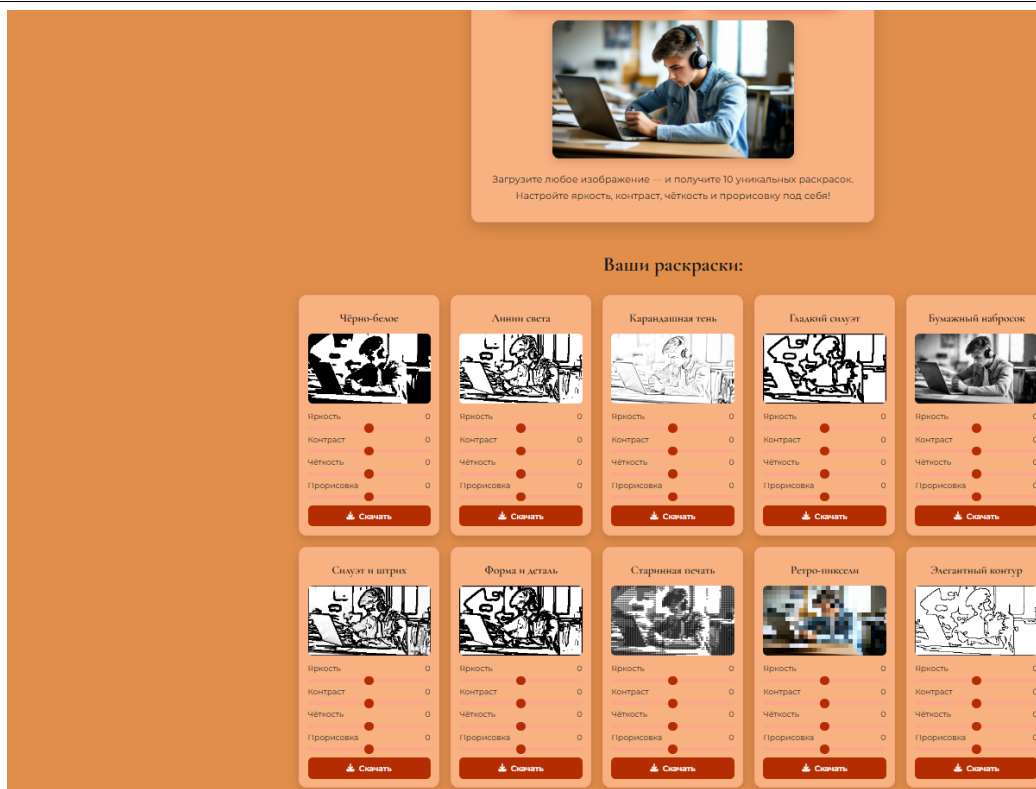


Рисунок 3 – Десять художественных стилей изображения

Описание функций стилей:

- 1) `applyThreshold` — использует простую пороговую бинаризацию. Каждый пиксель переводится в чёрный или белый в зависимости от яркости. Простой, но эффективный способ создания контрастной раскраски.
- 2) `applyEdgeDetect` — реализует оператор Собеля для выявления резких перепадов яркости. Результат — только контуры объектов, что идеально подходит для технических чертежей.
- 3) `applySketch` — классический алгоритм карандашного рисунка: изображение инвертируется, размывается, затем применяется режим наложения Color Dodge. Получается мягкий, естественный эффект.
- 4) `applyCleanOutline` — сначала бинаризует изображение, затем выделяет границы между чёрными и белыми областями и дополнительно утолщает линии для лучшей видимости.
- 5) `applyPencilSketch` — создаёт серый рисунок и добавляет случайный шум, имитирующий текстуру бумаги. Подходит для реалистичных набросков.
- 6) `applyContourSketch` — комбинирует результаты `applySketch` и `applyEdgeDetect`: фон остаётся мягким, а контуры подчёркнуты чёрными линиями.
- 7) `applyContourEdges` — объединяет два типа границ: внешние (силуэт) и внутренние (детали), что даёт богатую структуру для раскрашивания.
- 8) `applyEngraving` — оригинальный алгоритм: для каждой точки

рассчитывается яркость, и в зависимости от неё рисуется от 0 до 4 горизонтальных штрихов. Имитирует старинную печатную графику.

9) applyPixelArt — уменьшает изображение в 8 раз, затем масштабирует обратно с блочной интерполяцией. Создаёт ностальгический 8-bit стиль.

10) applyThinLine — аналог applyEdgeDetect, но с последующей очисткой шумов и оставлением только тонких, чётких линий без заливки.

Все функции работают на уровне пикселей через ImageData и CanvasRenderingContext2D, что обеспечивает высокую точность и совместимость.

Одной из ключевых особенностей веб-приложения «Художник: Раскрась Мир» является возможность тонкой настройки визуальных параметров для каждого из 10 стилей независимо (рис. 4). После генерации раскраски пользователь может корректировать четыре параметра с помощью ползунков под каждым изображением:

- 1) Яркость (от -100 до +100)
- 2) Контраст (от -100 до +100)
- 3) Чёткость (от -100 до +100)
- 4) Прорисовка (от -100 до +100)

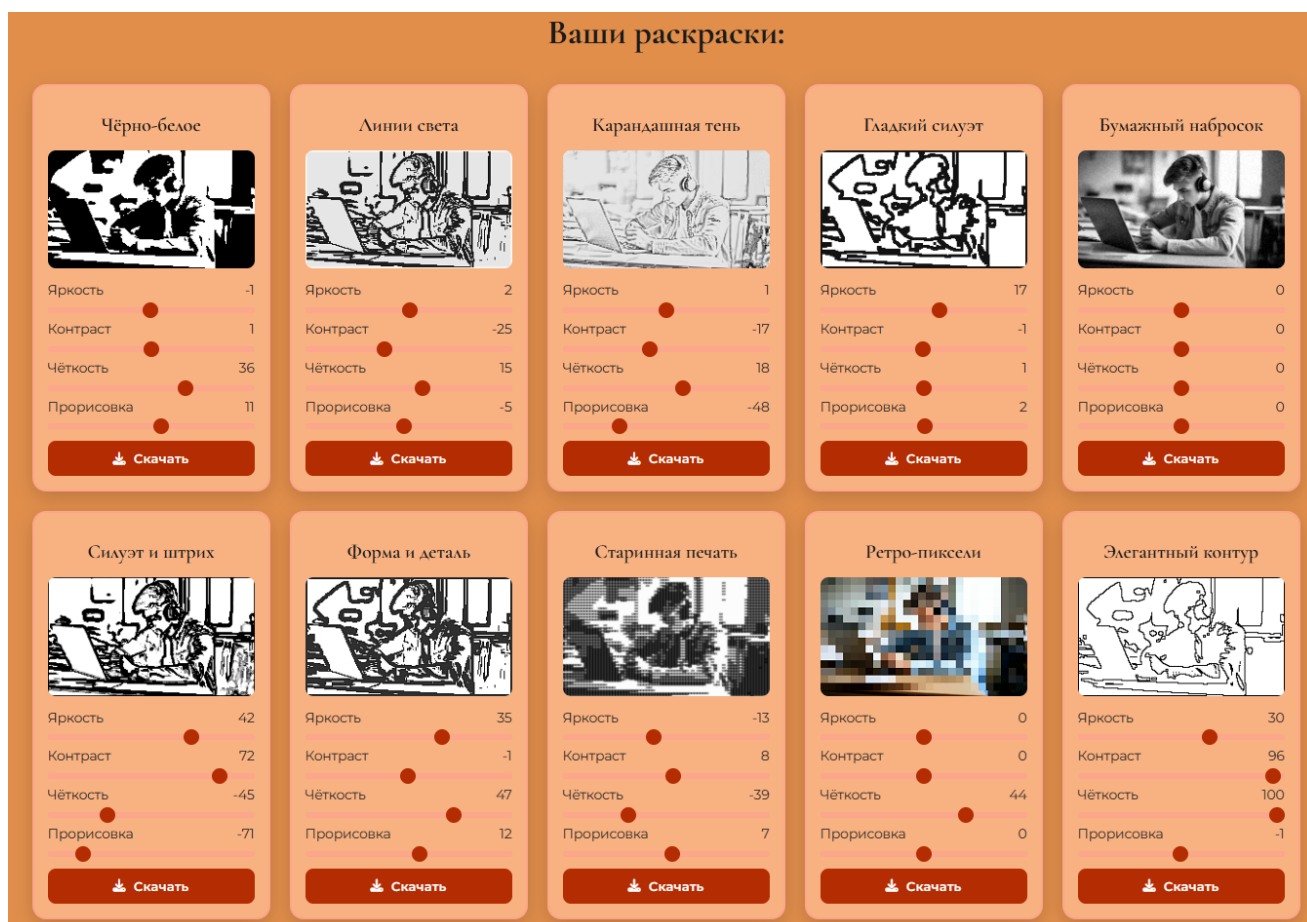


Рисунок 4 – Настройки визуальных параметров изображений

Эти параметры не влияют на исходный художественный стиль, а применяются поверх него к уже обработанному изображению, что позволяет сохранить уникальную семантику стиля, но адаптировать его под конкретные задачи — например, сделать контуры более чёткими для детей младшего возраста или затемнить тени для художественного эффекта.

Все изменения реализованы в функции `applyAdjustments(entry)`, которая вызывается при каждом движении любого из ползунков. При этом оригинальные пиксельные данные (`originalData`) сохраняются в объекте `entry` и используются как основа для всех пересчётов, что исключает накопление ошибок при многократной настройке.

Пример: применение яркости и контраста

Формула для одновременного применения яркости и контраста выглядит следующим образом:

```
const contrastFactor = (259 * (contrast + 255)) / (255 * (259 - contrast));
for (let i = 0; i < data.length; i += 4) {
  let r = orig[i];
  let g = orig[i + 1];
  let b = orig[i + 2];
  let a = orig[i + 3];
  r = contrastFactor * (r - 128) + 128 + brightness;
  g = contrastFactor * (g - 128) + 128 + brightness;
  b = contrastFactor * (b - 128) + 128 + brightness;
```

`brightness` — значение яркости (в диапазоне ± 100),

`contrast` — значение контраста (в диапазоне ± 100),

`contrastFactor` — стандартный коэффициент усиления контраста по формуле из теории обработки изображений.

Пример: регулировка чёткости

Чёткость реализована через упрощённый алгоритм Unsharp Mask:

```
if (sharpness > 0) {
  const amount = sharpness / 100;
  const blurred = getBlurredPixel(orig, i, entry.width, entry.height);
  r = r + (r - blurred.r) * amount;
  g = g + (g - blurred.g) * amount;
  b = b + (b - blurred.b) * amount;
} else if (sharpness < 0) {
  const amount = -sharpness / 100;
  const blurred = getBlurredPixel(orig, i, entry.width, entry.height);
  r = r - (r - blurred.r) * amount;
  g = g - (g - blurred.g) * amount;
  b = b - (b - blurred.b) * amount;}
```

При положительных значениях (`sharpness > 0`) изображение становится резче за счёт добавления разности между оригиналом и размытым вариантом. При отрицательных — мягче (эффект размытия).

Функция `getBlurredPixel()` вычисляет среднее значение цвета в окрестности 3×3 пикселя, что обеспечивает локальное сглаживание.

Пример: параметр «Прорисовка»

Параметр «Прорисовка» управляет контрастом между светлыми и тёмными областями с акцентом на тени:

```

if (drawing !== 0) {
    const gray = 0.299 * r + 0.587 * g + 0.114 * b;
    const darkAmount = drawing / 100;
    if (gray < 128) {
        r = r * (1 - darkAmount);
        g = g * (1 - darkAmount);
        b = b * (1 - darkAmount);
    } else {
        r = r + (255 - r) * darkAmount;
        g = g + (255 - g) * darkAmount;
        b = b + (255 - b) * darkAmount;
    }
}

```

При положительных значениях тени становятся глубже, а светлые области — ярче, что имитирует ручную штриховку карандашом. При отрицательных — изображение «сглаживается», теряя детализацию.

После обработки изображения и, при необходимости, индивидуальной настройки параметров пользователь может сохранить результаты локально на своём устройстве. Приложение поддерживает два режима экспорта:

1. Скачивание одного изображения (рис.5).

Под каждым стилем расположена кнопка «Скачать». При её нажатии вызывается метод `canvas.toDataURL('image/png')`, который преобразует содержимое соответствующего `<canvas>` в строку формата Data URL. Далее создаётся временный HTML-элемент `<a>` с атрибутом `download`, содержащим семантическое имя файла (например, `raskraska-karandashnaya-ten.png`), и программно эмулируется клик по этой ссылке. Это позволяет мгновенно сохранить изображение без перезагрузки страницы или обращения к серверу.

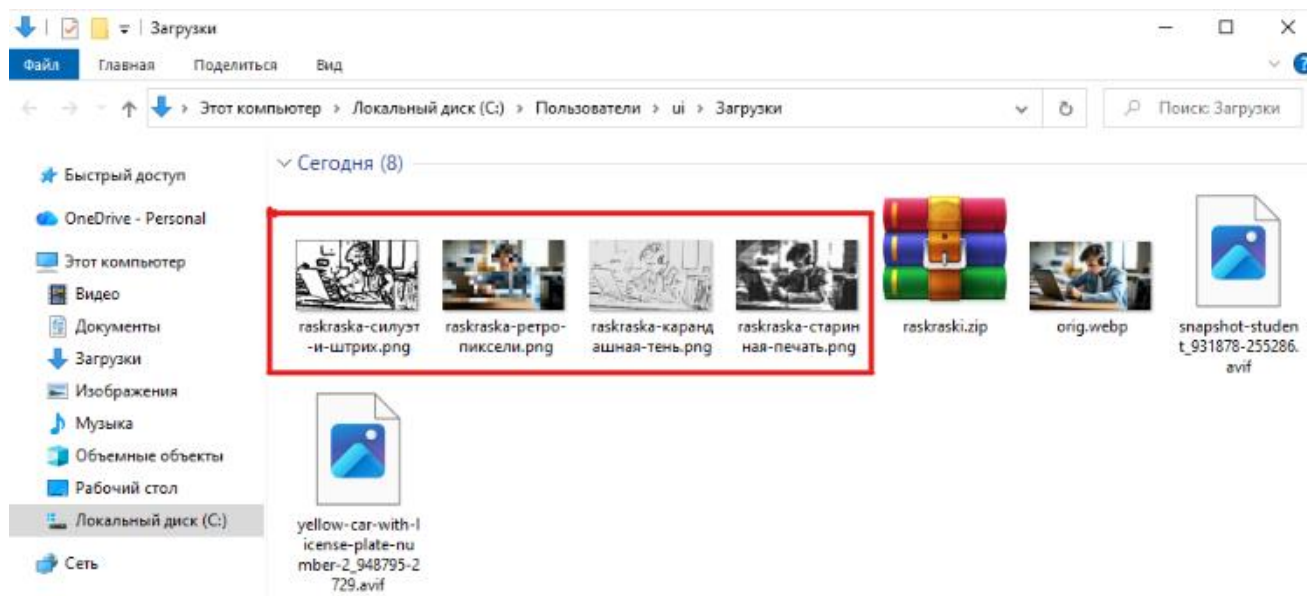


Рисунок 5 – Скачивание одного изображения

2. Скачивание всего набора (рис.6).

Кнопка «Скачать всё (ZIP)», активируемая после загрузки исходного изображения, запускает пакетное сохранение всех десяти раскрасок. Каждый `<canvas>` последовательно преобразуется в PNG через `toDataURL()`, извлекается base64-часть строки, которая затем передаётся в объект JSZip.

Библиотека формирует ZIP-архив с внутренней папкой raskraski, куда помещаются все файлы с понятными названиями. Готовый архив конвертируется в Blob и сохраняется на устройство пользователя с помощью функции saveAs() из библиотеки FileSaver.js.

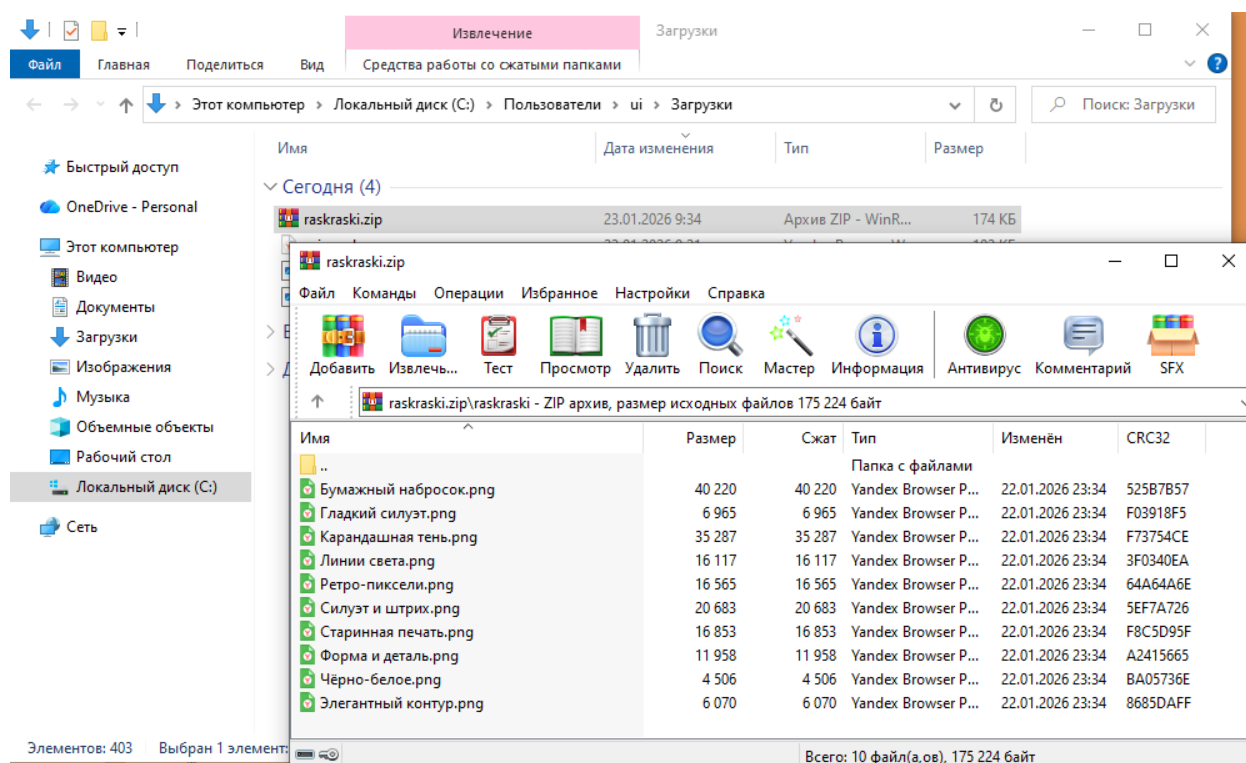


Рисунок 6 – Скачивание всего набора

Весь процесс экспорта реализован полностью на стороне клиента: данные никогда не покидают браузер, не передаются на внешние серверы и не требуют подключения к интернету после первоначальной загрузки страницы. Это обеспечивает высокий уровень конфиденциальности, мгновенную скорость работы и совместимость с любыми устройствами, поддерживающими современные веб-стандарты.

4 Выводы

В ходе исследования было успешно разработано веб-приложение «Художник: Раскрась Мир», которое демонстрирует широкие возможности нативных веб-технологий для создания профессиональных графических инструментов. Благодаря модульной архитектуре, использованию Canvas API и внешних библиотек для экспорта, приложение сочетает визуальную выразительность, функциональность и удобство. Оно может применяться как в образовательных целях (изучение обработки изображений, веб-разработки), так и в быту — для создания раскрасок, дизайнерских макетов или художественных экспериментов. Все компоненты работают автономно, не зависят от внешних сервисов и легко адаптируются под дальнейшее расширение.

Библиографический список

1. Аксенова С. А. Разработка веб-сайта на основе HTML, CSS и JavaScript // Студент года 2025: сборник статей Международного учебно-исследовательского конкурса, Петрозаводск, 10 марта 2025 года. Петрозаводск: Международный центр научного партнерства «Новая Наука» (ИП Ивановская И.И.), 2025. С. 102-109.
2. Белый Ю. В., Жадан М. И. Разработка веб-сайта «Фристайл. Лыжная акробатика» с использованием HTML 5, CSS 3, JavaScript // Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях: Материалы XXVII Республиканской научной конференции студентов и аспирантов (Гомель, 18–20 марта 2024 года). Гомель: ГГУ им. Ф. Скорины, 2024. С. 162.
3. Дакетт Д. HTML и CSS. Разработка и дизайн веб-сайтов. ЛитРес, 2018.
4. Козловская Е. В., Кузьменков Д. С. Разработка веб-сайта для тестирования по основам программирования с использованием HTML5, CSS3, JavaScript // Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях: Материалы XXVIII Республиканской научной конференции студентов и аспирантов (Гомель, 17–19 марта 2025 года). В двух частях. Часть 2. Гомель: ГГУ им. Ф. Скорины, 2025. С. 67.
5. Ефремова И. Н., Петрик Е. А., Кириллов А. М. Программно-информационная система для конструирования веб-сайтов // Известия Юго-Западного государственного университета. 2025. Т. 29. №. 3. С. 171-181.
6. Веб-приложения «Художник: Раскрась Мир» URL: <https://drive.google.com/file/d/1vcp9TgmeJraXE5vvGWKNG1jK67MNHINZ/view?usp=sharing> (дата обращения: 23.01.2026)
7. FileSaver.js URL: <https://github.com/eligrey/FileSaver.js> (дата обращения: 23.01.2026)
8. JSZip URL: <https://stuk.github.io/jszip/> (дата обращения: 23.01.2026)