## Содержание

[Содержание 2](file:///C:\Users\fedr\Desktop\paint\отчеты\отчет.docx#_Toc46146414)

[Введение 3](file:///C:\Users\fedr\Desktop\paint\отчеты\отчет.docx#_Toc46146415)

[Архитектура 3](file:///C:\Users\fedr\Desktop\paint\отчеты\отчет.docx#_Toc46146416)

[Сложности в проекте 4](file:///C:\Users\fedr\Desktop\paint\отчеты\отчет.docx#_Toc46146417)

[Удачные решения в проекте 5](file:///C:\Users\fedr\Desktop\paint\отчеты\отчет.docx#_Toc46146418)

[Неудачные решения в проекте 7](file:///C:\Users\fedr\Desktop\paint\отчеты\отчет.docx#_Toc46146419)

[Заключение 9](file:///C:\Users\fedr\Desktop\paint\отчеты\отчет.docx#_Toc46146420)

[Компетенции 10](file:///C:\Users\fedr\Desktop\paint\отчеты\отчет.docx#_Toc46146421)

[Список литературы 11](file:///C:\Users\fedr\Desktop\paint\отчеты\отчет.docx#_Toc46146422)

[Приложение 12](file:///C:\Users\fedr\Desktop\paint\отчеты\отчет.docx#_Toc46146423)

## Введение

Создание простейшего графического редактора – мой проект по летней практике. Данный проект был выбран в связи с тем, что он показался интересным и содержательным так как в нем были возможности для :

1. Различных реализаций модулей(что было наиболее интересно)
2. Практики в браузерного API.
3. Работы с интересной и перспективной технологией.
4. Усиления знаний в Javascript.
5. Улучшения знаний в рисовании простейшей графики.

Этими цели были руководящими при выборе проекта для летней практики.

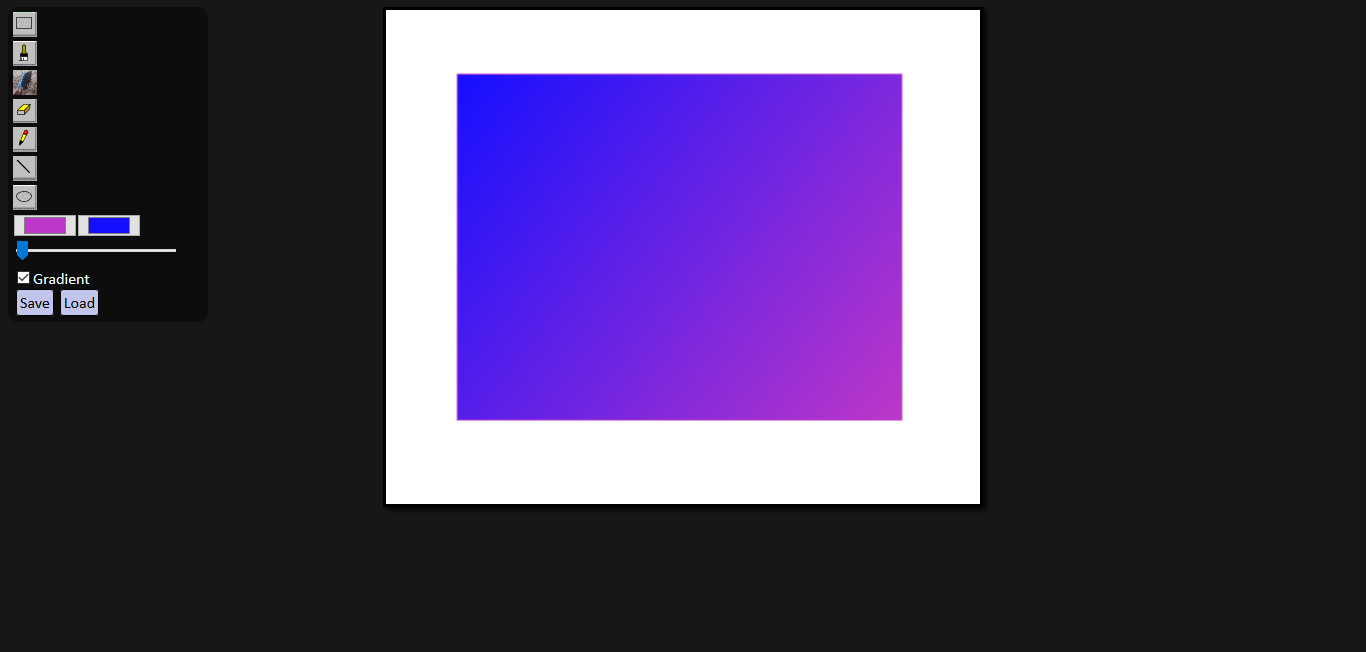


Рисунок 1 - пример готового приложения

Далее будет проведен обзор некоторых элементов из разработки данного проекта.

## Архитектура

В данной реализации простейшего графического редактора представлены несколько модулей : Loader, Tools, Presenter, Controller, ImageIO. Каждый модуль ответственен за свою часть логики. Начнем с начала:

1. Loader – загрузчик подключаемый в самом HTML файле. Его часть логики это загрузка всех основных модулей и выставка основных настроек и общих данных. Концептуальная задача – создать базовое состояние.
2. Tools – важнейший модуль хранящий всю логику используемых инструментов. Концептуальная задача – хранить реализацию всех инструментов и предоставлять эту информацию для других модулей.
3. Presenter – модуль работающий напрямую с DOM-деревом, динамически добавляя в него все недостающие элементы. Все изменения в данном модуле. Концептуальная задача – представить всю логику в HTML.
4. Controller – модуль добавляющий к документу все события. Концептуальная задача – отвечать за изменения состояния системы.
5. ImageIO –небольшой модуль, который отвечает за предоставление интерфейса сохранения и загрузки файлов.

Итак, в самом начале Loader выставляет базовое состояние для самого Canvas(получает его по id и подключает контекст), далее в первую очередь загружается модуль Tools, в котором реализована логика работы событий для каждого отдельного инструмента, а также каждый элемент Tools хранит свой id и путь к картинке, которая будет описывать данный инструмент. После этого загружается Presenter который добавляет в HTML все соответствующие блоки (инструменты а также другие контролирующие элементы вместе с их картинками). В это же время подгружается Controller, который будет отвечать за изменения глобального состояния(какой должен быть включен инструмент, какой цвет и т.д.). И заключительный элемент это ImageIO с его логикой загрузки и выгрузки картинок.

Далее будут описаны сложности, которые были встречены в данном проекте.

## Сложности в проекте

До начала работы с данным проектом мое знакомство с Javascript было поверхностным, аналогичные были знания и понимание работы DOM-дерева. Во время работы было улучшено владение связкой HTML + CSS + JS, а также повышено понимание взаимосвязей этих технологий. При осуществлении практической работы были получены начальные знания о событийной модели, о как принципах взаимодействия JS и HTML, о браузерном API и о логике, скрывающейся за оконными менеджерами. Также хочется добавить о неудавшихся, но важных элементах : при попытке реализации собственного ColorPicker-а было получено понимание аналогичности механизмов рисования в canvas собственных графических элементов и работы системных графических блоков. Данные темы объединяют схожие алгоритмы обработки данных:

1. Рассчитать на какую область был сам клик
2. Провести событие связанное с этим кликом
3. Произвести обязательные внутренние изменения.
4. (пассивно) Предоставить API взаимодействия с данным блоком.

Вместе с тем при выполнении проекта была получена хорошая возможность узнать больше о DOM-дереве и методах работы с ним(структуры данных, функции, методы добавления и удаления элементов).

Это позволяет открыть для себя еще одну возможность использования canvas – тренировка в создании своей графической оболочки и разного рода движков. Более того canvas благодаря своему интерфейсу позволяет создавать любого рода графические объекты и анимации. С помощью похожих методов в данном блоке вполне реально создать полностью самостоятельный видеопроигрыватель или же даже базовый видеоредактор. Canvas – очень эффективен и предоставляет качественный интерфейс для разработчика сайтов, человека желающего попрактиковаться в графике или же даже ученого, которому требуется визуализация(некоторые из анимаций на YouTube-канале 3Blue1Brown сделаны как раз в canvas, что лишний раз подтверждает удобство данной технологии).

Кроме того, на этом проекте появилась возможность больше узнать о самом JS. В первую очередь хочется заметить – JS это динамический язык программирования, поэтому тут нет такой явной системы классов(тут вообще не совсем классы) как в других языках. В самом начале возникли большие трудности из-за необычности языка и непонимания того, каким образом нужно проектировать с помощью JS. Вопросы возникали даже во время подключения файлов. В JS, для подключения файла к самому HTML-документу требуется динамически создать в HTML соответствующий блок и уже туда добавить нужный модуль, на начальных этапах данный способ вызывал некоторые осложнения. Далее большие трудности вызвало отсутствие си-подобного глобального и локального контекста(такого же явного как в С/С++), данные трудности были преодолены, но они несмотря на это оказали сильное влияние на конечную реализацию. Несмотря ни на что, самую большую проблему вызвало необычное взаимодействие между объектами и сам прием программирования – Прототипно-Ориентированное Программирование. На некоторое время данная особенность языка вызвала затруднние в разработке продукта. После продолжительных раздумий данное затруднение было разрешено созданием нужных для данного проекта объектов. Был реализован конструктор, делающий нужные для решения задачи шаблонные объекты, например, “Tools”(в сущности это был тот же класс, но в данном случае это не имело серьезного значения). После понимания концепции динамических классов(по своей сути объекты эквивалентны динамическим классам) осложнений в проекте было гораздо меньше.

Также хочется привести примеры успешных решений в рамках выполнения поставленных целей.

## Удачные решения в проекте

В данном проекте обнаружилось достаточное количество удачных на мой взгляд решений:

1. Создание объекта EventWrapper данный объект имеет 1 свойство и 1 метод : eventName – имя события( например, «mousedown»), и eventHandler – функция реализующая данное событие. Такой необычный объект позволяет хранить в себе все что нужно для подписки на событие с помощью функции addEventListener и соответственно позволяет полностью инкапсулировать логику работы с событиями в какой-либо объект.
2. Создание объекта Tool – это объект содержащий в себе минимум 2 свойства : id(данное свойство будет использоваться при добавлении данного объекта в HTML), imgSource – путь к картинке, которая будет отображаться на панели инструментов. А также данный объект содержит неограниченное количество объектов EventWrapper, которые инкапсулируют события обрабатываемые данным инструментом.

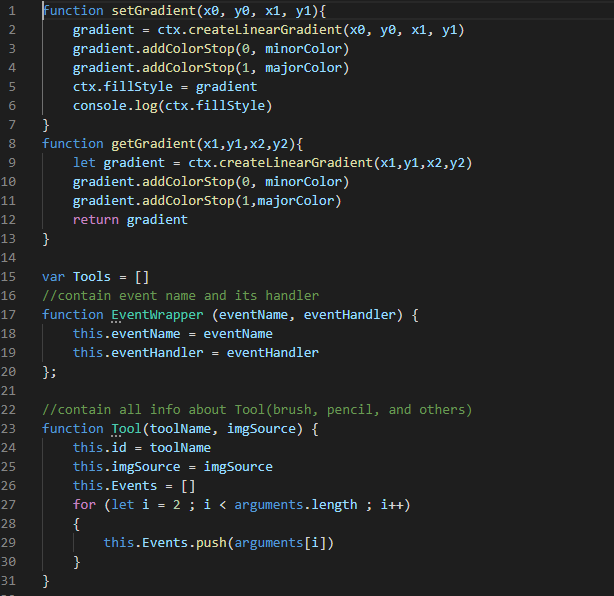


Рисунок 2 - пример успешного решения

1. Способ добавления нового инструмента : нужно всего лишь добавить в массив Tools нового элемента – не требуется менять ни один другой блок, именно этого и требовалось добиться при данной реализации.
2. Создание загрузчика и вынесение логики проверки стартового состояния в отдельный модуль. Данное решение позволило полностью отдельно развивать данный модуль и при добавлении в него любой последующей логики не требуется менять каким-либо образом другие модули.

## Неудачные решения в проекте

Однако в данном проекте были и случаи с неправильными решениями :

1. Ручное дописывание всех input-полей и их нестандартизированность. Чтобы дописать каждое input-поле требуется залезть в 2 модуля : Presenter, в котором прописать его отображение в HTML-документ, и в Controller, в котором прописать его событийную модель. То есть, чтобы добавить какой-то новый элемент нужно залезть в 2 модуля – это принципиально неправильно, более того некоторые из этих элементов требуют особую стилизацию, что тоже архитектурно неверно.



Рисунок 3 - пример неудачного решения

1. Не использование namespace-ов(одних больших объектов) и, поэтому порождение кучи лишних переменных находящихся в глобальном контексте, что очень сильно может повредить дальнейшей разработке.
2. Слишком отсутствие межмодульных интерфейсов. В модулях также отсутствуют внешние интерфейсы и, хотя по сути своей везде используется лишь массив Tools и ctx, а также в Controller используются функции из ImageIO, это все равно не приемлимо.
3. Отсутствие отдельного модуля настроек, который бы реализовывал изменение настроек всего приложения в одном определенном месте.
4. Отсутствие прокладки между API браузера и внутренней логикой. Наличие специального внутреннего модуля, являющегося прокладкой между браузером сильно облегчило бы разработку в дальнейшем.

Это основные плюсы и минусы, которые хотелось бы отметить.

## Заключение

В данном проекте были поставлены цели, связанные с практикой Javscript, практикой работы с браузерным API а также практикой разбиения программы на модули. Можно с уверенностью сказать, что данные цели успешно выполнены, хотя есть и недостатки, из которых были сделаны выводы : внутренняя логика всегда должна работать с внутрипрограммными абстракциями, то есть всегда должна быть прослойка между любым блоком, от которого зависит проектируемая система, всегда надо строить явные программные интерфейсы, если же для каких-то двух модулей их требуется слишком много, то возможно эти модули стоит объединить или же перепроектировать, так как иначе они в любом случае «срастутся» вместе, в Javascript надо делать как можно меньше свободных переменных, надо стараться, чтобы каждая переменная находилась в каком-то объекте, который логически должен ее содержать, если при добавлении новой логики требуется переписать весь модуль, то вероятно внутри модуля есть какой-то подмодуль, который стоит вынести отдельно. Тем не менее были получены фундаментальные знания Javascript и освоены навыки Прототипно-Ориентированного Программирования на языке Javascript. Кроме того были углублены и усилены знания в области верстки и использования таких технологий как CSS и HTML.Вместе с тем был получен важнейший опыт разработки приложения с использованием событий и событийной модели, а также взаимодействия с ними через использования языка Javascipt.Также в данном задании были изучены основные методы работы с такой технологией как canvas и было получено основополагающее понимание создания графических элементов, так как при помощи canvas можно нарисовать и наделить логикой абсолютно любой логический элемент, вплоть до собственного оконного менеджера внутри браузера. Также были получены навыки в самостоятельном поиске нужной информации.

## Список литературы

## Мартин, Роберт К. Чистый код[Текст]/Мартин Роберт К. - Санкт-

Петербург: Питер, 2010 - 464 с.

1. Мартин, Роберт К. Чистая архитектура[Текст]/Мартин Роберт К. -

Санкт-Петербург: Питер, 2018 - 352 с.

1. Флэнаган, Д. Javascript для начинающих[Текст]/ Флэнаган Д. – Москва:

Символ-Плюс, 2008 – 992 с.

1. Макфарланд, Дэвид С. Новая большая книга CSS[Текст]/Макфарланд

Дэвид С. – Санкт-Петербург: Питер, 2018 – 720 с.

1. Бхаргава, Адитья Грокаем алгоритмы[Текст]/Бхаргава Адитья – Санкт-Петербург: Питер, 2019 – 288 с.