МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики»

**Мегафакультет компьютерных технологий и управления**

**Факультет программной инженерии и компьютерной техники**

**Направление (специальность) – 09.04.02 Информационные системы и технологии**

**Специализация –Веб-технологии**

**Курсовой проект по дисциплине**

**Проектирование и анализ языков веб-решений**

**ТЕМА:** Исследование методологий автоматизации разработки веб-приложений на основе клиентских и серверных фреймворков

ВЫПОЛНИЛ

Студент группы P4162 Оболенский И.И. № группы подпись, дата ФИО

ПРОВЕРИЛ к.п.н., доцент \_\_\_\_\_\_\_\_\_\_ Государев И.Б.

ученая степень, подпись

должность

САНКТ-ПЕТЕРБУРГ

2019 г.

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,**

**МЕХАНИКИ И ОПТИКИ»**

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | | Оболенский Илья Игоревич | | | | | | |
| (Фамилия, И., О.) | | | | | | | | |
| Факультет | ПИиКТ | | | | | | Группа | P41622 |
| Направление (специальность) | | | | | | 09.04.02 «Информационные системы и технологии» | | |
| Руководитель | | | | | Государев И.Б., Университет ИТМО, доцент, к.пед.н. | | | |
| (Фамилия, И., О., место работы, должность, ученое звание, степень) | | | | | | | | |
|  | | | | | | | | |
| Дисциплина | | | | Проектирование и анализ языков веб-решений | | | | |
|  | | | | | | | | |
| Наименование темы: | | | Исследование методологий автоматизации разработки веб-приложений на основе клиентских и серверных фреймворков | | | | | |

Задание

Произвести сравнительный анализ средства автоматизации сборки веб-приложений.

Краткие методические указания

Исследовать особенности наиболее популярных средств автоматизации сборки веб-приложений. Произвести сравнительный анализ наиболее популярных средств сборки веб-приложений. Разработать приложение с использованием различных средств сборки веб-приложений.

Содержание пояснительной записки

Оглавление. Введение. Ход выполнения работы – иллюстрированное описание языка, грамматики, библиотек, фреймворков, действий и применяемых программ, команд, приёмов, параметров, а также обоснование используемых средств, моделей, параметров. Заключение. Список использованной литературы.

Рекомендуемая литература

1. Пьюривал С. Основы разработки веб-приложений.—СПб.: Питер, 2015.—272 с.

2. В.И. Бойков, С.В. Быстров, А.С. Кремлев, К.А. Сергеев. Правила оформления курсовых и квалификационных работ. — СПб: СППб ГУ ИТМО, 2007. – 39 с.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Государев И.Б.   
 (подпись) (Фамилия И.О.)

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оболенский И.И.  
 (подпись) (Фамилия И.О.)

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

**ГРАФИК КУРСОВОГО ПРОЕКТА (РАБОТЫ)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | | Оболенский Илья Игоревич | | | | | |
| (Фамилия, И., О.) | | | | | | | |
| Факультет | ПИиКТ | | | | | Группа | P41622 |
| Направление (специальность) | | | | | 09.04.02 «Информационные системы и технологии» | | |
| Руководитель | | | | Государев И.Б., Университет ИТМО, доцент, к.пед.н. | | | |
| (Фамилия, И., О., место работы, должность, ученое звание, степень) | | | | | | | |
|  | | | | | | | |
| Дисциплина | | | Проектирование и анализ языков веб-решений | | | | |
|  | | | | | | | |
| Наименование темы: | | | | Исследование методологий автоматизации разработки веб-приложений на основе клиентских и серверных фреймворков | | | |
|  | | | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Наименование этапа | Дата завершения | | Оценка и подпись руководителя |
| Планируемая | Фактическая |
| 1. | Исследование особенностей наиболее популярных средств автоматизации сборки веб-приложений. | Апрель 2019 |  |  |
| 2. | Сравнительный анализ наиболее популярных средств сборки веб-приложений. Разработка приложения с использованием различных средств сборки веб-приложений. | Май 2019 |  |  |

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Государев И.Б.

(подпись) (Фамилия И.О.)

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оболенский И.И.

(подпись) (Фамилия И.О.)

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

**АННОТАЦИЯ К КУРСОВОМУ ПРОЕКТУ (РАБОТЕ)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | | Оболенский Илья Игоревич | | | | | |
| (Фамилия, И., О.) | | | | | | | |
| Факультет | ПИиКТ | | | | | Группа | P41622 |
| Направление (специальность) | | | | | 09.04.02 «Информационные системы и технологии» | | |
| Руководитель | | | | Государев И.Б., Университет ИТМО, доцент, к.пед.н. | | | |
| (Фамилия, И., О., место работы, должность, ученое звание, степень) | | | | | | | |
|  | | | | | | | |
| Дисциплина | | | Проектирование и анализ языков веб-решений | | | | |
|  | | | | | | | |
| Наименование темы: | | | | Исследование методологий автоматизации разработки веб-приложений на основе клиентских и серверных фреймворков | | | |
|  | | | | | | | |

ХАРАКТЕРИСТИКА КУРСОВОГО ПРОЕКТА (РАБОТЫ)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1. Цель и задачи работы** | | Предложены студентом | | Определены руководителем | |
| Цель работы — сравнительный анализ клиентских сборщиков веб-приложений.  Задачи работы: | | | | | |
| 1. Обзор особенностей наиболее популярных сборщиков веб-приложений. | | | | | |
| 1. Сравнение результатов сборки проекта различными сборщиками. | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **2. Характер работы** | | | | | |
|  | | | | | |
| Расчет | Конструирование | | Моделирование | | Другое |
| **3. Содержание работы**  Произведен анализ наиболее популярных сборщиков приложений Webpack, Rollup и Parcel, | | | | | |
| произведено сравнение сборщиков приложений Webpack, Rollup и Parcel на примере | | | | | |
| разработки приложения Калькулятор. | | | | | |
|  | | | | | |
| **4. Выводы** | |  | | | |
| Требования к проекту реализованы | | | | | |
|  | | | | | |

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Государев И.Б.   
 (подпись) (Фамилия И.О.)

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оболенский И.И.

(подпись) (Фамилия И.О.)

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

**ОТЗЫВ РУКОВОДИТЕЛЯ**

**о выполнении курсового проекта (работы)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | | Оболенский Илья Игоревич | | | | | | |
| (Фамилия, И., О.) | | | | | | | | |
| Факультет | ПИиКТ | | | | | | Группа | P41622 |
| Направление (специальность) | | | | | | 09.04.02 «Информационные системы и технологии» | | |
| Руководитель | | | | Государев И.Б., Университет ИТМО, доцент, к.пед.н. | | | | |
| (Фамилия, И., О., место работы, должность, ученое звание, степень) | | | | | | | | |
|  | | | | | | | | |
| Дисциплина | | | Проектирование и анализ языков веб-решений | | | | | |
|  | | | | | | | | |
| Наименование темы: | | | | | Исследование методологий автоматизации разработки веб-приложений на основе клиентских и серверных фреймворков | | | |
| |  | | --- | |  | | | | | | | | | |
|  | | | | | | | | |

**ОЦЕНКА КУРСОВОГО ПРОЕКТА (РАБОТЫ)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Показатели** | **Оценка** | | | |
| **п/п** | **5** | **4** | **3** | **0** |
| 1. | Проект создан обучающимся самостоятельно |  |  |  |  |
| 2. | Предложенные выводы обоснованы |  |  |  |  |
| 3. | Проект технологически грамотный |  |  |  |  |
| 4. | Оформление отвечает требованиям к отчету, отчёт размещён в веб-портфолио |  |  |  |  |
| 5. | Во время защиты обучающийся показал умение кратко, доступно представить результаты работы, умение анализировать, аргументировать свою точку зрения, делать обобщение и выводы, адекватно ответить на поставленные вопросы. |  |  |  |  |
| **ИТОГОВАЯ ОЦЕНКА** | |  | | | |

**Отмеченные достоинства:**

|  |
| --- |
| творческая активность при выполнении работы, грамотный анализ и обобщение |
| информационного материала. В процессе выполнения НИР выполнены все поставленные |
| задачи. Научный стиль изложения, содержание логически не противоречиво |
|  |

**Отмеченные недостатки:**

|  |
| --- |
| необходимо расширить поле анализируемых публикаций по теме исследования, |
| добавить подробное описание конфигурационных файлов средств автоматизации |
| сборки |
|  |
|  |

**Заключение:**

|  |
| --- |
| Обучающийся подтвердил владение компетенциями, предусмотренными учебным |
| планом. |
|  |

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Государев И.Б.   
 (подпись)

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019

Оглавление

[Введение 8](#_Toc10208900)

[Сборщики веб-приложений 10](#_Toc10208901)

[Webpack 10](#_Toc10208902)

[Rollup 11](#_Toc10208903)

[Parcel 13](#_Toc10208904)

[Сравнение результатов сборки веб-приложения различными сборщиками 15](#_Toc10208905)

[Сервер веб-приложения 16](#_Toc10208906)

[Сравнение сборок веб-приложения 16](#_Toc10208907)

[Заключение 19](#_Toc10208908)

[Список использованных источников 21](#_Toc10208909)

# Введение

В современном мире веб-разработка является одним из самых востребованных направлений программирования. Большинство компаний от небольших до самых крупных имеют свой сайт. В настоящее время активно развивается направление конструкторов сайтов, таких как Readymag, Tilda, Wix, они позволяют быстро создавать шаблонные сайты, но имеют ограниченный функционал. Чтобы привлекать клиентов, студиям веб-разработки необходимо предоставлять им выгодные условия и конкурировать с конструкторами сайтов, для этого требуется оптимизировать время и качество разработки, автоматизировать процесс разработки.

Выполнение описанных выше требований достигается путем использования клиентских и серверных фреймворков, которые обладают следующими преимуществами в сравнении с созданием сайтов с использованием языков программирования без дополнительных инструментов и средств автоматизации разработки:

1. Структурирование файлов для крупных приложений, что позволяет ускорить процесс нахождения нужных файлов.
2. Повторное использование кода за счет создания компонентов и классов.
3. Большое количество готовых решений, компонентов и библиотек для популярных фреймворков.
4. Удобство масштабирования команды разработчиков, за счет использования известных и документированных стандартов.

Использование клиентских фреймворков осуществляется вместе со средствами автоматизации сборки проекта – приложениями, позволяющими собрать js и css компоненты в один файл для подключения на сайт[1].

В данном курсовом проекте рассматриваются средства автоматизации сборки проекта, как один из способов автоматизации разработки веб-приложений.

**Цель работы:** сравнительный анализ клиентских сборщиков веб-приложений.

**Задачи:**

1. Обзор особенностей наиболее популярных сборщиков веб-приложений.
2. Сравнение результатов сборки проекта различными сборщиками.

# Сборщики веб-приложений

Сборщики необходимы при разработке крупных веб-приложений, когда проект состоит из большого количество js и css файлов. На веб-страницу js-файлы можно подключать последовательно, при этом соблюдая определенный порядок, либо подключить один файл, который получится при сборке проекта. Подключение одного файла не только упростит разработку, но и ускорит загрузку страницы, так как будет выполнено меньшее количество http-запросов[2].

В качестве наиболее популярных сборщиков веб-приложений были выбраны Webpack, Rollup и Parcel. Каждый из данных сборщиков обладает определенными преимуществами, в ходе курсового проекта необходимо определить лучший из приведенных сборщиков.

## Webpack

Webpack является наиболее популярным из приведенных сборщиков и обладает следующими преимуществами:

1. Поддержка и развитие проекта.
2. Большое количество дополнительных функциональных возможностей, таких как загрузчики (loaders) и плагины.
3. Возможность создания отдельных кофигураций для различных сборок проекта[3].
4. Возможность запуска тестового сервера с использованием webpack-dev-server.
5. Использование пресетов для сборки проектов на основе фреймворков и библиотек таких как ReactJS.

Недостатки webpack:

1. Высокий порог входа для создания сложных конфигураций.
2. Недостаточно подробная документация.
3. Частое обновление проекта, за счет чего сложно найти актуальные обучающие уроки[4].

На рисунке 1 приведен пример конфигурационного файла webpack.config.js. Из примера видно, что простейший файл настроек для webpack является небольшим и его назначение понятно из названия настроек:

* entry – входной файл, откуда webpack начинает просматривать и собирать проект, entry может быть строкой, массивом или объектом;
* output – описание параметров для сборки проекта, path – путь директории для сборки, filename – название файла[5].



Рисунок - Пример конфигурационного файла webpack.config.js

## Rollup

Rollup – сборщик ES6-модулей, который преобразовывает их в отдельную сборку, являющуюся модулем в формате ES6 или CommonJS[6].

Rollup обладает следующими преимуществами:

1. Получаемая сборка состоит только из используемых экспортов благодаря технике tree-shaking.
2. Сборка ES6-модулей в ES6-формат.
3. Поддержка сборки модулей в IIFE-формат.
4. Небольшой размер собранного кода[7].

Недостатки Rollup:

1. Меньшее количество плагинов и менее распространен в сравнении с webpack и browserify.
2. Подробная документация[8].
3. Для сборки css-файлов необходимо использовать плагин.

На рисунке 2 приведен пример конфигурационного файла Rollup. Input – входной файл сборщика, output – выходной файл и формат сборки, plugins – подключаемые плагины.

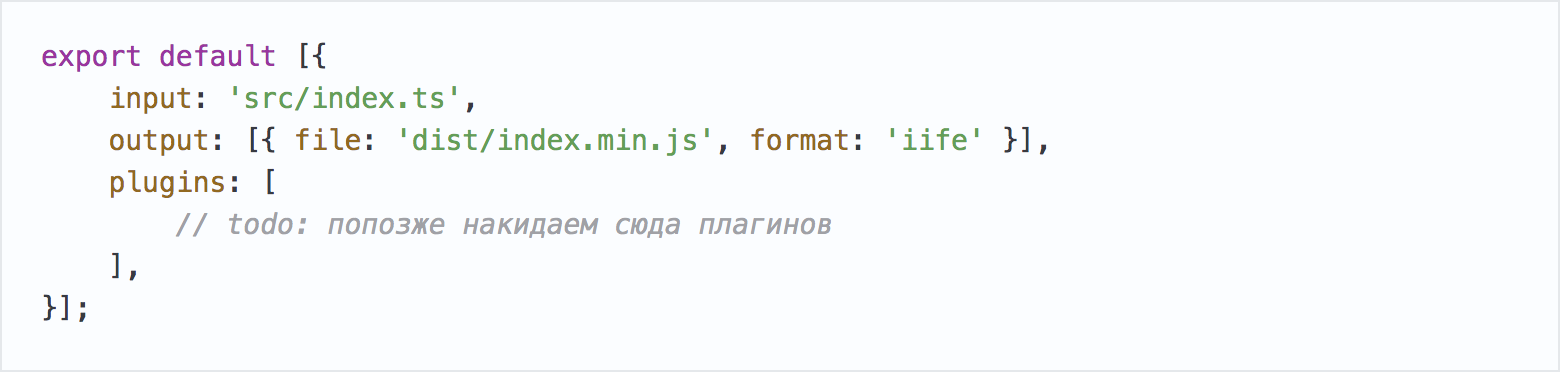


Рисунок - Пример конфигурационного файла Rollup

Простейшие конфигурации для webpack и rollup выглядят похоже. Но если требуется что-то более сложное, конфигурационный файл webpack разрастается и становится более сложным, а в Rollup необходимо только подключить нужные плагины (Рисунок 3).



Рисунок - Пример конфигурационного файла Rollup для сборки проекта веб-приложения с поддержкой подключения модулей commonjs, typescript, подгрузкой html и scss

## Parcel

Parcel – упаковщик веб-приложений, который может собирать приложения без дополнительных настроек, для сборки приложения Parcell требуется только указать путь до HTML-файла[9].

Преимущества Parcel:

1. Скорость сборки. Parcel использует воркеры для включения многоядерной компиляции и имеет кеш файловой системы для быстрых повторных сборок, в том числе после перезапуска сборки.
2. Поддержка сборки JS, CSS и HTML-файлов без дополнительных настроек.
3. Автоматическая трансформация кода с использованием Babel, PostCSS и PostHTML.
4. Автоматическое обновление модуля в браузере при его изменении в процессе разработки.
5. Логирование ошибок путем выделения ошибочного синтаксиса.
6. Возможность подключения дополнительных плагинов.

Parcel обладает следующими недостатками:

1. Parcel неудобен в использовании для настройки более сложных, нестандартных веб-приложений.
2. Требует больше вычислительных ресурсов от компьютерах[10].
3. Меньшее компьюнити, чем у Webpack[11].

Для запуска простейшей сборки Parcel не требуется конфигурационный файл. Запуск сборки Parcel происходит при помощи написания следующей команды в терминале: parcel index.html.

Если требуется подключение дополнительных плагинов, таких как PostCSS или PostHTML, необходимо создать дополнительные файлы postcss.config.js (Рисунок 4) и posthtml.config.js (Рисунок 5).

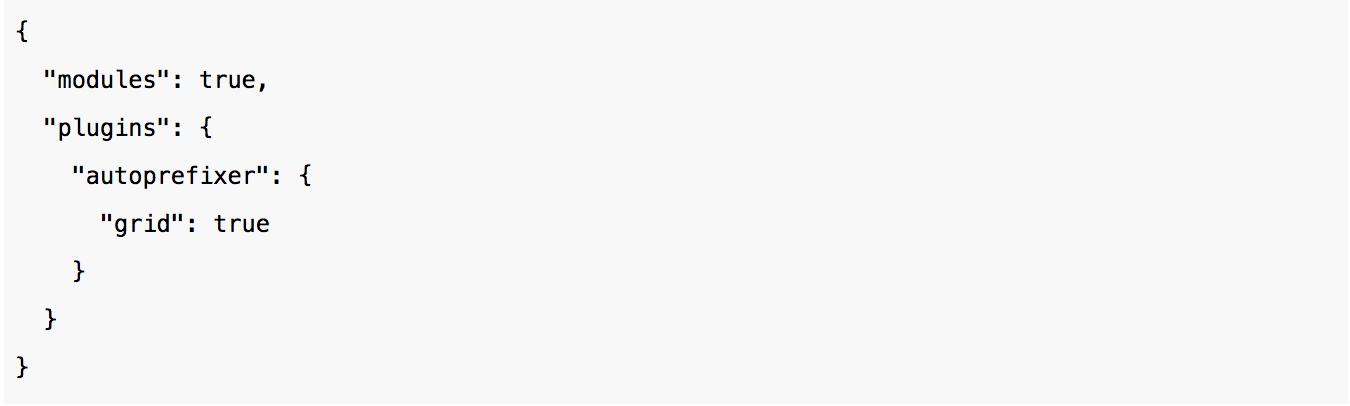


Рисунок – Пример конфигурационного файл postcss.config.js

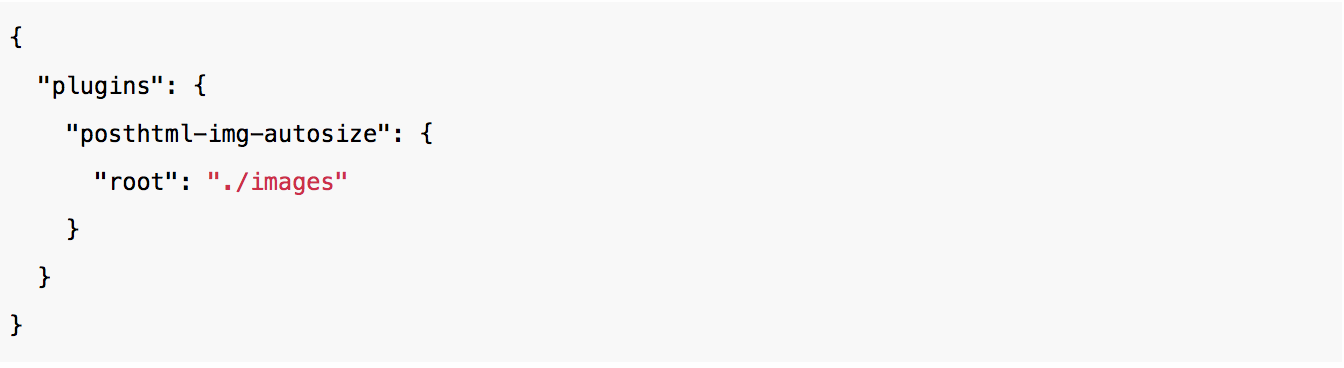


Рисунок - Пример конфигурационного файла posthtml.config.js

# Сравнение результатов сборки веб-приложения различными сборщиками

Для сравнения результатов сборки веб-приложения различными сборщиками были выбраны сборщики, приведенные в предыдущем разделе: Webpack, Rollup и Parcel.

В качестве приложения был выбран калькулятор, обладающий следующим функционалом:

1. Возможность ввода арифметического приложения с клавиатуры.
2. Возможность ввода знаков арифметических операций при помощи кнопок на экране приложения: “+”, “-“, “\*”, “/”, квадратный корень.
3. Вывод результата арифметического выражения в поле ввода выражения при клике на кнопку “=”.

Приложение должно соответствовать дизайну, представленному на рисунке 6.

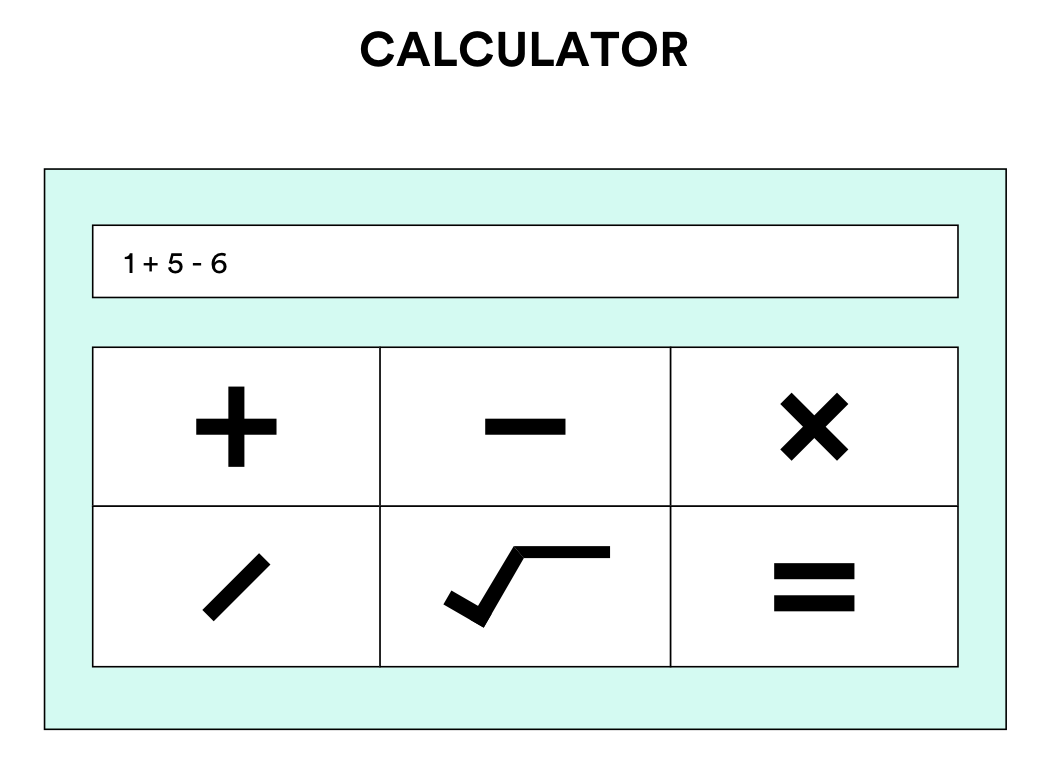


Рисунок - Дизайн приложения Калькулятор

## Сервер веб-приложения

В качестве сервера для разработки проекта на локальном компьютере был выбран express, как простой в настройке и подключении сервер [12].

Конфигурационный файл dev-сервера на express представлен на рисунке 7.



Рисунок - Конфигурационный файл dev-сервера express.js

Запуск сервера для каждого проекта с различной сборкой осуществляется при помощи вызова npm-команды npm run start, которая описана в разделе scripts конфигурационного файла package.json. Сервер запускается командой node dev/server.js.

## Сравнение сборок веб-приложения

Для сравнения сборщиков веб-приложений были выбраны следующие критерии:

1. Размер конфигурационного файла.
2. Количество требуемых npm-зависимостей.
3. Количество подключаемых плагинов.
4. Размеры собранных файлов.
5. Время сборки.

Результаты сравнения сборщиков веб-приложения приведены в таблице 1.

Таблица - Сравнение сборщиков веб-приложений

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий для сравнения | Webpack | Rollup | Parcel |
| Размер конфигурационного файла | 1 кб | 484 б | Не требуется конфигурационный файл |
| Количество требуемых npm-зависимостей | 14 | 6 | 4 |
| Количество подключаемых плагинов | 3 | 2 | 1 |
| Размеры собранных файлов | main.js – 8 кб  main.css - 1 кб | main.js – 1 кб  main.css – 1 кб | main.js – 14 кб  main.css – 1 кб |
| Время сборки | 527 мс | 109 мс | 481 мс |

Как видно из сравнительной таблицы для Parcel не требуется конфигурационный файл, для Webpack самый большой размер конфигурационного файла – 1 кб, то есть для webpack требуется больше настроек. Так же это следует из большего количества требуемых npm-зависимостей и большего количества подключаемых плагинов.

Тем не менее при внесении каких-либо изменений в стандартную сборку Parcel, приходится подключать дополнительные плагины. Некоторые настройки сборки не могут быть реализованы при помощи Parcel. Например, при помощи Parcel нельзя скопировать папку из одной директории в другую. Можно только скопировать содержимое папки с статичными файлами в папку static. Для этого требуется установить при помощи npm плагин plugin-parcel-asset-copier и добавить в package.json опцию “assetsPath”: “src/assets”.

Лучший результат сборки js-файла показал Rollup. Сборщик Rollup не добавляет никаких лишних скриптов в результирующий js-файл, кроме того позволяет выбирать режим сборки одной опцией настройки в конфигурационном файле. Webpack и Parcel добавляют в результирующий js-файл много дополнительных скриптов, что значительно увеличивает размер файла.

Так же Rollup выполняет сборку значительно быстрее в сравнении с Webpack и Parcel, что является следствием того, что Webpack и Parcel добавляют в результирующий файл дополнительные скрипты, увеличивая тем самым время сборки.

Таким образом, на основе приведенного сравнительного анализа сборщиков веб-приложений можно сделать вывод, что Rollup является наиболее оптимальным сборщиком. Он проигрывает Parcel тем, что для Parcel не требуется конфигурационный файл, и Parcel требуется меньшее количество npm-зависимостей и плагинов. Тем не менее, конфигурационный файл Rollup имеет понятную структуру, Rollup, в отличие от Parcel, хорошо документирован, что позволяет достаточно просто выполнять настройку сборки, а количество требуемых npm-модулей и используемых плагинов отличается незначительно. По приведенным выше параметром Rollup является более простым в настройке в сравнении с Webpack.

Rollup значительно выигрывает у обоих сборщиков Webpack и Parcel в производительности. Размер собранного файла Rollup и время сборки в несколько раз меньше в сравнении с Webpack и Parcel.

# Заключение

В результате выполнения курсового проекта был произведен анализ наиболее популярных сборщиков приложений Webpack, Rollup и Parcel и произведено их сравнение на примере сборки простого приложения Калькулятор.

На основе произведенного анализа можно сделать следующие выводы:

1. Webpack является наиболее популярным сборщиком веб-приложений, гибко настраиваемым, но недостаточно хорошо документирован, что усложняет настройку сборки с его помощью. Также webpack генерирует дополнительный код в результирующий js-файл, что увеличивает его размер и время сборки веб-приложения. Таким образом Webpack лучше всего подходит для сложных веб-приложений большого объема, для которых требуется специализированная настройка конфигурации.
2. Rollup менее популярен в сравнении с Webpack, но также имеет достаточно большое сообщество пользователей, хорошую документацию и прост в настройке. Размер конфигурационного файла значительно меньше, чем размер конфигурационного файла Webpack. Различные настройки Rollup можно задавать в конфигурационном файле, например, формат сборки js-файла: ES6, IIFE, CommonJS. Настройки, которые Rollup не поддерживает, можно подключать при помощи использования плагинов, которые в свою очередь также могут обладать настройками. Rollup является самым производительным сборщиком в сравнении с Webpack и Parcel, исходя и сравнения сборок на примере приложения Калькулятор.
3. Parcel хорошо подходит для простых стандартных приложений, не требующих дополнительных настроек, имеет документацию на русском языке и плагины, расширяющие его функционал. Parcel сравним с Webpack по скорости сборки и объему собранного js-файла, но если требуется подключение дополнительных нестандартных настроек, настройка сборки при помощи Parcel усложняется.

# Список использованных источников

1. Хабр. Ускоряем сборку веб-приложения с webpack [Электронный ресурс]. – URL: https://habr.com/ru/post/451146/ (дата обращения: 25.05.2019).
2. Medium. Введение в философию Webpack [Электронный ресурс]. – URL: https://medium.com/devschacht/aggelos-arvanitakis-an-introduction-to-webpack-s-philosophy-d12415cc8594 (дата обращения: 25.05.2019).
3. Хабр. 10 особенностей Webpack [Электронный ресурс]. – URL: https://habr.com/ru/company/plarium/blog/309230/ (дата обращения: 25.05.2019).
4. WebforMyself. Изучите Webpack за 15 минут [Электронный ресурс]. – URL: https://webformyself.com/izuchite-webpack-za-15-minut/ (дата обращения: 25.05.2019).
5. Tproger. Webpack: основы настройки проекта на JavaScript и Sass [Электронный ресурс]. – URL: https://tproger.ru/translations/webpack-basics/ (дата обращения: 25.05.2019).
6. Хабр. 6 впечатляющих веб-технологий 2015 года [Электронный ресурс]. – URL: https://habr.com/ru/company/plarium/blog/277485/ (дата обращения: 25.05.2019).
7. Хабр. Rollup: уже можно собирать приложения [Электронный ресурс]. – URL: https://habr.com/ru/post/440946/ (дата обращения: 25.05.2019).
8. Rollup.js. [Электронный ресурс]. – URL: https://rollupjs.org/guide/en (дата обращения: 25.05.2019).
9. Parcel.js. [Электронный ресурс]. – URL: https://parceljs.org/ (дата обращения: 25.05.2019).
10. LogRocket. Switching to Parcel from Webpack [Электронный ресурс]. – URL: https://logrocket.com/blog/switching-to-parcel-from-webpack/ (дата обращения: 25.05.2019).
11. Jakoblind. Parcel vs webpack [Электронный ресурс]. – URL: https://blog.jakoblind.no/parcel-webpack/ (дата обращения: 25.05.2019).
12. Express.js [Электронный ресурс]. – URL: https://expressjs.com/ (дата обращения: 25.05.2019).

**Приложение А**

**Конфигурационный файл webpack.config.js**

**const** *webpack* = require(**'webpack'**);  
**const** path = require(**'path'**);  
**const MiniCssExtractPlugin** = require(**"mini-css-extract-plugin"**);  
**const** CopyWebpackPlugin = require(**'copy-webpack-plugin'**);  
**const** UglifyJSPlugin = require(**'uglifyjs-webpack-plugin'**);  
  
**module**.**exports** = {  
 **entry**: [**'./src/app.js'**],  
 **module**: {  
 **rules**: [  
 {  
 **include**: [path.resolve(\_\_dirname, **'src/js'**)],  
 **loader**: **'babel-loader'**,  
  
 **options**: {  
 **plugins**: [**'syntax-dynamic-import'**],  
  
 **presets**: [  
 [  
 **'@babel/preset-env'**,  
 {  
 **modules**: **false** }  
 ]  
 ]  
 },  
  
 **test**: /\.js$/  
 },  
 {  
 **test**: /\.(less|css)$/,  
  
 **use**: [  
 {  
 **loader**: **MiniCssExtractPlugin**.**loader** },  
 {  
 **loader**: **'css-loader'**,  
  
 **options**: {  
 **sourceMap**: **true** }  
 },  
 {  
 **loader**: **'less-loader'**,  
  
 **options**: {  
 **sourceMap**: **true** }  
 }  
 ]  
 }  
 ]  
 },  
  
 **output**: {  
 **chunkFilename**: **'[name].[chunkhash].js'**,  
 **filename**: **'[name].js'**,  
 **path**: path.resolve(\_\_dirname, **'static'**)  
 },  
  
 **mode**: **'development'**,  
  
 **plugins**: [  
 **new MiniCssExtractPlugin**({  
 **filename**: **'main.css'** }),  
 **new** CopyWebpackPlugin([  
 {  
 **from**: **'src/svg'**,  
 **to**: **'svg'** }  
 ])  
 ]  
};

**Файл package.json для приложения со сборкой при помощи Webpack**

{  
 **"devDependencies"**: {  
 **"@babel/core"**: **"^7.4.4"**,  
 **"@babel/preset-env"**: **"^7.4.4"**,  
 **"babel-loader"**: **"^8.0.6"**,  
 **"babel-plugin-syntax-dynamic-import"**: **"^6.18.0"**,  
 **"css-loader"**: **"^2.1.1"**,  
 **"less"**: **"^3.9.0"**,  
 **"less-loader"**: **"^5.0.0"**,  
 **"style-loader"**: **"^0.23.1"**,  
 **"uglifyjs-webpack-plugin"**: **"^2.1.2"**,  
 **"webpack"**: **"^4.31.0"**,  
 **"webpack-cli"**: **"^3.3.2"**,  
 **"express"**: **"^4.16.2"**,  
 **"mini-css-extract-plugin"**: **"^0.5.0"**,  
 **"copy-webpack-plugin"**: **"^5.0.3"** },  
 **"scripts"**: {  
 **"build"**: **"webpack --watch --config webpack.config.js"**,  
 **"start"**: **"node dev/server.js"** }  
}

**Приложение Б**

**Конфигурационный файл rollup.config.js**

**import** lessModules **from 'rollup-plugin-less-modules'**;  
**import** copy **from 'rollup-plugin-copy'  
  
export default** [{  
 **input**: **'src/app.js'**,  
 **output**: {  
 **file**: **'static/main.js'**,  
 **format**: **'esm'** },  
 **plugins**: [  
 *// подключение less стилей* lessModules({  
 **output**: **'static/main.css'** }),  
 copy({  
 **targets**: [  
 **'src/svg'** ],  
 **outputFolder**: **'static'** })  
 ]  
}];

**Файл package.json для приложения со сборкой при помощи Rollup**

{  
 **"devDependencies"**: {  
 **"chalk"**: **"^2.4.2"**,  
 **"express"**: **"^4.17.0"**,  
 **"less"**: **"^3.9.0"**,  
 **"rollup"**: **"^1.12.3"**,  
 **"rollup-plugin-copy"**: **"^2.0.1"**,  
 **"rollup-plugin-less-modules"**: **"^0.1.11"** },  
 **"scripts"**: {  
 **"build"**: **"rollup -c rollup.config.js --watch"**,  
 **"start"**: **"node dev/server.js"** }  
}

**Приложение В**

**Файл package.json для приложения со сборкой при помощи Parcel**

{  
 **"devDependencies"**: {  
 **"chalk"**: **"^2.4.2"**,  
 **"express"**: **"^4.17.0"**,  
 **"less"**: **"^3.9.0"**,  
 **"parcel-plugin-asset-copier"**: **"^1.0.0"** },  
 **"scripts"**: {  
 **"build"**: **"parcel watch src/app.js --out-dir static --out-file main.js"**,  
 **"start"**: **"node dev/server.js"** },  
 **"assetsPath"**: **"src/assets"**}